indescomp

LAS 40 MEJORES SUBRUTINAS

EN CODIGO MAQUINA PARA EL SPECTRUM



LAS 40 MEJORES RUTINAS EN CODIGO MAQUINA PARA EL ZX SPECTRUM

(Con texto aclaratorio)

AUTORES:

JOHN HARDMAN
ANDREW HEWSON

TRADUCCION:

JESUS RUIZ ARRIBAS
Estación Espacial de Madrid (NASA-INTA)

Editado por **INDESCOMP, S.A.**
Pº de la Castellana, 179 - Madrid-16

*Todas las consultas relativas a este libro deberán ser dirigidas a **INDESCOMP, S.A.**

Imprime: **CONORG, S.A.**

I.S.B.N.: **84-86176-09-3**

Depósito Legal: **M-14590-1984**

PROLOGO

Este libro podría considerarse como el necesario complemento al manual del ZX Spectrum.

Para el no iniciado, este libro le aportará los conocimientos básicos que le ayuden a comprender la manera de trabajar del ZX Spectrum, a nivel de código máquina. Para el experto supondrá un inestimable libro de referencia. Tanto para uno como para el otro, las rutinas aportadas por los autores de este libro, suponen una notable mejora al desarrollo de los programas que el usuario implemente sobre su ZX Spectrum. Ciertas rutinas de la sección B, potencian en gran medida las prestaciones de dicho ordenador personal, aumentando su rendimiento y ampliando el campo de sus aplicaciones.

El estudio a fondo de este libro y la práctica con las rutinas expuestas, llevará al usuario a perfeccionar sus programas y a sacarle mayor partido a la impresora y a los gráficos en pantalla.

También es de destacar la mejora que introducen en el ZX Spectrum, algunas rutinas dedicadas a ejecutar efectos especiales sobre la pantalla, tales como mezcla de imágenes (incluyendo color), agrandamiento de zonas de la imagen, rotación de caracteres, etc.

Si bien el comprender el funcionamiento de las rutinas de la sección B puede parecer arduo y difícil, han de pensar que la primera parte del libro puede resultar clave para esa tarea. Ciertas rutinas se pueden adaptar con bastante facilidad al ZX81, sobre todo si se tiene un buen nivel de conocimientos de lenguaje de máquina del microprocesador Z80A.

Aunque a lo largo de la traducción ha prevalecido el espíritu de fidelidad al idioma español, éste resulta poco flexible para expresar concisamente la jerga técnica anglosajona y no ha habido por menos que mantener ciertas palabras en su idioma original para no aumentar el volumen y grado de dificultad del presente libro.

Jesús Ruiz Arribas

CONTENIDO

Sección A

Sección B

Sección A

1. INTRODUCCION

El propósito de este libro es dotar, tanto al principiante como al usuario de computadores ya experimentado, de una fuente de referencia, siempre a mano, con un cierto número de rutinas en código máquina que van a ser las más usuales, interesantes y hasta entretenidas, para el ZX Spectrum. La sección "A" describe las propiedades del Spectrum que son de interés para quien programe en código máquina —que comprende las propiedades internas más importantes del ZX Spectrumm así como la estructura del lenguaje máquina.

La sección "B" nos presenta las rutinas propiamente dichas. Estas están tratadas aparte en forma normalizada y se explican con detalle al principio de la sección. Las rutinas están completas, de modo que se pueden cargar individualmente sin hacer referencia a ninguna otra.

En principio no se necesita saber cómo trabaja una rutina para hacer uso de ella debido a que cada una se puede cargar usando un sencillo "Cargador M/C" (M/C Loader), cuyo listado se puede encontrar al principio de la sección "B". Sin embargo, si usted está impaciente por usarlas, solo nos resta decir que se dirija, por ejemplo, a la página donde está la rutina llamada "Listar Variables" (List Variables), introduzca y "Ponga en Marcha" (RUN) el "Cargador M/C" (M/C Loader) y a continuación introduzca los números decimales listados en la columna cuya cabecera se titula "Números a introducir". Cuando todos los números estén cargados compare el valor dado por "La Prueba de la Suma" (Check sum) que ha impreso el cargador M/C, con el valor dado por la rutina. Si coinciden, puede estar seguro que los números introducidos son correctos (salvo que haya tenido dos o más errores que se anulen entre sí exactamente). Recuerde que dicha "Prueba de la Suma" consiste en obtener la suma de los números decimales introducidos. La rutina, pues, está lista para su uso.

Si al principio no tiene la confianza suficiente como por ejemplo "Listar Variables", pero aún está interesado en iniciarse en código máquina tan pronto como le sea posible, elija entonces una rutina más corta. De esta manera, si se pierde o comete muchos errores no empleará mucho tiempo en corregirlos. La rutina ideal para esto puede ser "Listado de cualidades en sentido descendente" (Scroll Attributes Down). De nuevo, es una simple cuestión de introducir y "Poner en Marcha" (RUN) el "Cargador M/C" listado al principio de la Sección "B" y copiar los números de la columna encabezada por "Números a Introducir". Cuando haya terminado, asegúrese que "La Prueba de la Suma" es correcta.

Si usted se anima a postergar un poco el uso inmediato de las rutinas en código máquina, entonces lea atentamente el libro. El resto de este capítulo hace una introducción acerca de las ideas básicas y continúa con la explicación más en detalle de la estructura del resto del libro. Para el no iniciado es recomendable leer cuidadosamente esta información. Respecto al usuario más experimentado, podría requerir sólo leerlo superficialmente.

El microprocesador Z80A que maneja al ZX Spectrum, no entiende directamente las palabras escritas en BASIC, como puede ser: PRINT ("Impri-

mir"), IF ("Si-condicional"), TAB ("Tabular"), etc. En lugar de esto, el Z80A obedece a un lenguaje especial y propio llamada "código máquina" (Machine Code). Las instrucciones contenidas en la ROM (Memoria de Solo Lectura)[1] son las que confieren al Spectrum su "personalidad" y están escritas en ese leguaje especial. Consisten, a su vez, en un gran número de rutinas que introducen, listan, interpretan y ejecutan un dialecto particular de BASIC que usa el Spectrum. En efecto, las rutinas son grupos de instrucciones del tipo "Qué hacer si . . ." (What to do if . . .). Estas le dicen al Z80A "Qué hacer si el comando siguiente, es BASIC, es la palabra PRINT ("Imprimir") y luego "Qué hacer si", el siguiente elemento es un nombre de variable; y "Luego qué hacer si", el siguiente es una coma, etc.

El código máquina consiste en una secuencia de números enteros y positivos, cada uno menor de 256, que indican la acción a tomar al Z80A, emplazando ocho interruptores de acuerdo con el modelo que se obtiene al sustituirlo por el número equivalente en binario.(2)

El equivalente binario del numero 237, por ejemplo, es 11101101, de modo que cuando el número 237 sea encontrado por el Z80A, los ocho interruptores serán emplazados de modo siguiente: "cerrado, cerrado, cerrado, abierto, cerrado, cerrado, abierto y cerrado" respectivamente(3).

Esto se debe a que la máquina trabaja con la versión binaria de un número, por lo que no es necesario considerar de esta forma una instrucción para el entendimiento humano.

Nosotros estamos acostumbrados a trabajar en base decimal, y es de esta forma como la reconoce el "Cargador M/C" (M/C Loader) descrito en la Sección "B".

Sin embargo, en el caso de una "Sucesión" (String) de números decimales es dificultoso interpretarlos de modo que éstos se convierten de nuevo a un lenguaje especial llamado "Lenguaje de Ensamblamiento" (Assembly Language) el cual es un poco críptico, pero con la práctica no es demasiado difícil usarlo. Cada rutina descrita en la sección "B" es listada en forma numérica decimal y en "lenguaje de ensamblamiento".

---

(1) N. del T.: Referirse a la nota (1) del Capítulo 2 y al Epígrafe "Octetos" (BYTES) del Capítulo 3.

(2) N. del T.: Esta nota va dirigida principialmente al no iniciado. El modelo gráfico normalmente utilizado para representar el "0" ó el "1" empleado en el sistema binario es el de un interruptor eléctrico cuya posición de encendido (ON) es equivalente al "1" (es decir, hay paso de corriente, por lo tanto el contacto está cerrado) y la posición de apagado es equivalente al "0" (no hay paso de corriente, luego el contacto está abierto). Abierto y Cerrado son términos que emplearemos cuando hagamos referencia al "0" ó al "1", respectivamente.

(3) N. del T.: Hemos establecido el equivalente del "0" y el "1" como un interruptor abierto o cerrado. Desde luego que el Z80A no trabaja con interruptores propiamente dichos; en su lugar emplea impulsos eléctricos (provenientes de un oscilador o reloj) cuya presencia o ausencia corresponde a los estados lógicos "1" ó "0" respectivamente. La razón de emplear ocho interruptores en nuestra comparación viene dada por la longitud de "palabra" que maneja el microprocesador Z80A (8 bits). Un Bit es la mínima cantidad de información con la que trabaja un computador.

Se llama "Lenguaje de Ensamblamiento" porque un programa especial denominado "Ensamblador" puede ser convenientemente usado de manera que agrupe o ensamble muchas instrucciones dadas en código máquina para formar un programa nuevo. Debido a que el código máquina es muy extenso, los "ensambladores" son programas sofisticados que usualmente están escritos también en código máquina. Algunos de estos ensambladores estarán disponibles para el Spectrum, y toda vez que las rutinas descritas en este libro pueden ser cargadas usando un "ensamblador", hace que el "Cargador M/C" siga siendo bastante adecuado para este propósito.

Solo se requiere un número para especificar la instrucción más sencilla del Z80A.

Por ejemplo, la instrucción que "transfiere" o "copia" (COPY) el contenido del registro *c* al registro *d* es 81 en decimal (el significado de la palabra registro será explicada con más detalle en el capítulo 3). De momento es suficiente decir que *c* y *d* presentan una analogía con las variables BASIC. Para estas instrucciones existe una correspondencia "uno a uno" entre el número decimal y la versión del "lenguaje de ensamblamiento" de modo que el número decimal 81, por ejemplo, se escribe en dicho lenguaje como ld d, c. "ld" significa "cargar" (LOAD).

Muchas instrucciones del "Lenguaje de Ensamblamiento" consisten en similares abreviaciones y por esta razón se las llama nemónicos.

Las instrucciones más complejas requieren dos, tres y hasta cuatro números antes de estar completamente especificadas, en cuyo caso un simple nemónico de ensamblador bastará para representar todo. La tabla 1.1. da listado de unos cuantos ejemplos de números, sus nemónicos correspondientes y una breve explicación de los mismos.

Ref.	*Decimal*	*Leng. Ensamblador*	*Comentario*
(a)	81	ld d, c	Carga *d* con el contenido de *c*
(b)	14 27	ld c, 27	Carga *c* con el número 27
(c)	14 13	ld c, 13	Carga el número 13 en *c*
(d)	33 27 52	ld hl, 13339	Carga 13339 en la pareja de registros *hl* Adviértase que el número 13339 se obtiene con la siguiente operación: 27 + 256* 52 = 13339; 27 es cargado en el registro *l;* 52 es cargado en el *h*
(e)	221 33 27 52	ld hl, 13339	Carga 13339 en el registro dual "ix"

Tabla 1.1 *Ejemplos de instrucciones en código máquina para el Z80A*

La línea (a) de la tabla es un ejemplo de la instrucción discutida anteriormente. Las líneas (b) y (c) nos muestran cómo un número entero y positivo menor que 255 se puede cargar en un registro usando dos números consecuti-

vos. El primero de ellos especifica la acción a ejecutar y el segundo especifica el número a cargar. La línea (d) muestra cómo un número entero grande puede ser cargado en dos registros, *h* y *l*, juntos. En este caso son el segundo y el tercer número los que especifican el número a cargar. El ejemplo final de la línea (e) ilustra un código de cuatro números (grupos de números) usados para cargar un número entero grande en el registro dual ix. Hay que hacer notar que tres de los cuatro números también aparecen en la línea (d). En efecto, el primer número, especifica el registro dual ix en vez del registro dual hl.

La estructura del lenguaje máquina se explica con más detalle en el capítulo 3, y una lista completa de los nemónicos de ensamblador se da en el apéndice "A". Llegados a este punto, la pregunta más importante que nos podemos formular es:

¿Por qué usar el código máquina?

Con cualquier lenguaje de cualquier computador, ocurre siempre que existen tareas que el usuario quiere que ejecute el computador y que no pueden ser escritas convenientemente en el lenguaje de programación que se tenga a mano, o en el caso de estarlo es muy lenta su ejecución. El ZX Spectrum no es una excepción al respecto.

Consideremos por ejemplo el problema de guardar el contenido reflejado en la pantalla en su zona alta, en la memoria RAM(4) o quizá transferir desde la RAM a la pantalla con la intención de crear un efecto de "dibujo animado" mediante el "golpeteo" entre varias imágenes. El área de memoria ocupada por el "archivo de información gráfica en pantallla" (Display File) y el correspondiente a "Cualidades" (Attributes), ocupa 6.912 "octetos" (Bytes) (Palabra de 8 bits de longitud), de modo que es necesario mover o cambiar el tope de memoria (RAMTOP = Límite superior de memoria RAM), llevándolo al valor: 32768 — 6912 = 25856 para el modelo de 16 K(5), con el fin de proveer suficiente espacio a la transferencia, fuera del área reservada al BASIC (en el modelo de máquina de 48 K(5) será 65536 — 6912 = 58624).

A continuación se indica un sencillo programa en BASIC que deberá guardar en memoria la información contenida en la pantalla, aunque esto le tomará algún tiempo relativamente largo, de unos 70 segundos aproximadamente:

```
10 FOR i = 0 to 6911
20 POKE 25856 + i, PEEK (16384 + i)
30 NEXT i
```

---

(4) N. del T.: RAM es un término referido al tipo de memoria usado por el computador. Las siglas corresponden al acrónimo de las palabras inglesas RANDOM ACCESS MEMORY o Memoria de Acceso Aleatorio, es decir, los datos pueden ser introducidos o extraídos de la memoria sin buscar la posición en ésta de manera secuencial. Actualmente su fabricación es con tecnología MOS (METAL OXIDE SEMICONDUCTOR) y las más comunes son de dos tipos: Dinámicas y Estáticas. Las memorias dinámicas necesitan ser "refrescadas' cada pocos milisegundos (son las que emplea el Spectrum) y son de muy bajo consumo. Las de tipo estático no necesitan de "refresco" y consumen bastante más.

(5) N. del T.: Cuando se hace referencia a 16 K o 48 K se debe entender esta cifra como área libre de memoria y accesible por el usuario. 1 K octeto (Byte) = 1.024 octetos (Bytes).

La razón de ello es que el Spectrum emplea la mayor parte del tiempo en decodificar las órdenes antes de ejecutarlas. También convierte los números emplea en convertir los números a la forma de dos palabras de 8 bits cada una (octeto), que es la que el Z80A comprende. También convierte los números formando cinco palabras de 8 bits (octeto), que es el formato con el que trabaja el "contador" decimal en "lazo cerrado" (Loop Counter). Y por último, en ejecutar operaciones aritméticas con cinco palabras de 8 bits. La secuencia de ejecución es la siguiente:

1) Suma "i" a 16384.
2) Convierte el resultado a la forma de dos palabras de 8 bits.
3) "Recoge" el contenido de la dirección de la memoria expresada por la sentencia BASIC "PEEK".
4) Suma "i" a 25856.
5) Convierte el resultado en la forma de dos palabras de 8 bits.
6) "Almacena" el valor resultante en la dirección de memoria expresada por la sentencia BASIC "POKE".
7) Suma 1 al valor de "i" y almacena el resultado.
8) Resta "i" de 6911. Si el resultado es positivo o cero, vuelve al apartado 1).

Cada vez que recorre el lazo (desde la sentencia 1.ª a la 8.ª) el Spectrum debe decodificar cada "orden" previamente, ya que éste no retiene en memoria la operación efectuada con anterioridad.

Es fácil comprobar que el computador emplea cerca del 99% del tiempo preparando la tarea que ha de ejecutar, en vez de ejecutarla en sí estrictamente.

Ahora, ya no nos sorprende decir que una rutina en código máquina ejecuta la acción de guardar la información contenida en la pantalla más o menos instantáneamente. En la sección "B" se da un ejemplo de una rutina de este tipo.

¿Cómo aprender Código Máquina?

El lenguaje máquina del microprocesador Z80A es muy complejo y para comprender todas sus facultades se requiere un buen libro de consulta, una considerable cantidad de reflexión y mucha práctica. Existen varios libros disponibles, entre ellos podemos destacar: *"How to program the ZX80"*, que lo podemos considerar como una referencia básica, cuyo autor es Rodney Zaks y publicado por Sybex (disponible en cualquier establecimiento Radio Shack, perteneciente a la cadena Tandy Stores) con la referencia ISBN n.° 0-89588-057-1. Dicho libro contiene una gran cantidad de información acerca de la organización del "soporte físico" o circuitería (Hardware) del microprocesador, así como un listado completo y detallado del "juego de instrucciones" (Instruction Set). Quizá el no iniciado podría encontrarlo demasiado arduo porque tiene alrededor de 600 páginas. Es posible que resulte menos complicado leer el contenido del libro que lleva por título *"Z80 and 8080 Assembly Language Programming"*, cuyo autor es Kathe Spracklen, publicado por Hayden, con la referencia ISBN n.° 0-8104-5167-0. El libro comienza a un nivel más elemental y cubre los aspectos más importantes en cuanto a

''programación'' (Software) e ignora casi por completo lo relativo al ''soporte físico'' o circuitería (Hardware).

Este libro está orientado a servir no solo como introducción al código máquina para el no iniciado, sino también de utilidad para el usuario más experimentado. Esto, confiere al lector un fuerte incentivo para aprender código máquina proveyéndole de rutinas con las que pueda incorporar sus propios programas en BASIC o en código máquina, con o sin adaptaciones.

Muchas de las rutinas dependen íntimamente de la estructura del ZX Spectrum y así, el siguiente capítulo va a tratar el tema con más detalle. Cubrirá, por supuesto, la forma en que se trata el ''archivo de información gráfica'' (Display File), el ''área de programa'' (Program Area) y el ''área de variables'' (Variables Area). También explicará el esquema de las líneas que componen un programa en Basic, además de una introducción a las operaciones aritméticas con ''coma flotante'' (Floating Point) en la forma, ya comentada, de cinco octetos. Suponemos que el lector de la Sección ''B'' está familiarizado con el contenido de este capítulo.

El tercer capítulo tratará de explicar el lenguaje máquina del Z80A con cierto detalle, descubriendo muchos puntos que describimos más tarde. Estos contienen un glosario del ''juego de instrucciones'' (Instruction Set) que abarcarán a la mayoría de las más importantes sin llegar a la cobertura en detalle que proporciona el libro de R. Zaks.

2. ESTRUCTURA INTERNA DEL ZX SPECTRUM

Un computador es una máquina capaz de almacenar una secuencia de instrucciones y ejecutarlas después. Como es obvio, para hacer esto se requiere una memoria en la que puedan almacenarse las instrucciones. El ZX Spectrum contiene dos tipos distintos de memoria. El primero de ellos corresponde a la "memoria de solo lectura" (ROM)[1].

Esta memoria ROM contiene un conjunto de instrucciones implantadas en la máquina por el fabricante. El segundo tipo es la memoria de acceso aleatorio RAM[(2)].

La memoria de acceso aleatorio es el "borrador" (en el sentido de cuaderno de anotaciones) del Spectrum. Cuando el computador ejecuta una tarea, está continuamente "mirando" el contenido de la RAM, es decir, "leyendo" y también "alterando" su contenido o, lo que es igual, "escribiendo".

El Spectrum no hace uso del "borrador" al azar. Se usan diferentes partes de la RAM. Para almacenar diferentes clases de información. Por ejemplo, un programa escrito en Basic e introducido por el usuario será almacenado en la RAM en un lugar concreto, mientras que las variables que usa el programa son depositadas en otro lugar de la memoria RAM. El tamaño de ese "borrador" (Notepad) está limitado y por eso la máquina tiene mucho cuidado de ocupar el espacio que la información necesita y no más. De esta manera el espacio restante está siempre recogido en un lugar de modo que si el usuario quiere añadir una línea a su programa, la información en la RAM puede ser entremezclada usando algunos de los espacios sobrantes para dar acomodo a la línea extra.

La mayor parte de este capítulo está dedicada a explicar con detalle cómo el ZX Spectrum organiza la RAM debido a que muchas de las rutinas descritas en la sección "B" están diseñadas para manipular la RAM. Por otra parte, si el lector quiere comprender el diseño de las rutinas en vez de usarlas a ciegas, debera entender el contenido de este capítulo.

El capítulo cubre:

— El "Archivo de Información" (Display File).
— El "Archivo de Cualidades" (Attributes File).
— La "Memoria Intermedia" o "Memoria Tampón" de la impresora (Printer Buffer).
— Las "Variables del Sistema" (System Variables).

La sección final describe las rutinas alojadas en la ROM que se refieren a la sección "B".

---

(1) N. del T.: ROM es una abreviatura consistente en escoger la primera letra de las palabras inglesas READ ONLY MEMORY. Es un tipo de memoria cuyo contenido es permanente e imborrable y solo accesible por el ciclo de lectura de un microprocesador. Su uso, normalmente, es para almacenar las rutinas que manejan el sistema y para poner en marcha el computador cuando se le aplica la tensión de alimentación (BOOTSTRAP). En adelante la escribiremos siempre asi: ROM.

(2) N. del T.: Ver capítulo 1, nota 4.

Mapa de la memoria u organización de la memoria

Existen dos versiones del ZX Spectrum: la versión básica consta de 16384 posiciones de memoria de libre disposición por el usuario. La versión ampliada tiene ademaś otras 32768 posiciones adicionales, lo que hace un total de 49152 posiciones de acceso libre por el usuario. Cada posición puede contener un número entero simple cuyo valor estará comprendido entre 0 y 255 inclusive. La posición y por consiguiente el número entero que contiene, está identificada por una dirección, la cual se expresa mediante un número entero y positivo.

Las direcciones que van desde la 0 a la 16383 están asignadas de forma fija a la memoria ROM, de modo que la primera dirección asignada a la RAM será la 16384. La tabla 2.1 nos muestra el "Mapa de Memoria" del ZX Spectrum. Vemos en él cómo usa la RAM comenzando por la dirección 16384. Por ejemplo, el "Archivo de Información Gráfica" (Display File) que contiene en cada momento la información mostrada en pantalla, ocupa las posiciones que van desde la 16384 hasta la 22527.

Las "cualidades" (Attributes), que determinan el color, brillo, etc., de la Información en Pantalla sigue de inmediato la última posición asignada al "archivo de información gráfica"; es decir comenzando en la posición 22528 y terminando en la 23295 inclusive.

Las cinco primeras direcciones de comienzo (o "direcciones de arranque") expresadas en la columna 1.ª de la tabla 2.1 son todas ellas fijas, ya que el "archivo de información gráfica", las "cualidades", etc., ocupan una cantidad fija de espacio. La quinta área está asignada al mapa de la microunidad de discos (Microdrive). Si una unidad de este tipo está conectada al Spectrum se hallará en dicha área toda la información relativa a la estructura de los datos almacenados en la mencionada microunidad de discos.

En el caso en que la unidad no esté conectada, el área citada no es necesaria, por lo tanto, la sexta área pasará a ocupar el lugar de la quinta yendo inmediatamente después de la cuarta. El área que ocupa el sexto lugar está dedicada a los "Información de Canales". Este ahorro de espacio está en línea con la estructura del ZX Spectrum que trata siempre de no desperdiciar espacio mientras que sea posible. Así tendremos que la dirección de comienzo (o arranque) del área de "Información de Canales" y de todas las áreas que la siguen no son fijas, sino flotantes arriba y abajo de la memoria RAM.

El Spectrum tiene conocimiento en todo momento de la dirección de comienzo de todas las áreas, almacenando el valor actualizado de cada dirección dentro del área de "Variables del Sistema". Dicho área de "Variables del Sistema" se halla antes que la que corresponde a la "Información de la Microunidad de Discos", es decir, entre las posiciones 23552 y 23733 inclusive. De esta manera no existe problema ya que este área no se mueve arriba y abajo de la meoria RAM. La dirección de las áreas flotantes se enumera en la columna 2.ª de la tabla 2.1. Por ejemplo, la dirección de comienzo del área que ocupa el programa escrito en Basic está contenida en la posición 23635, dentro del área de "Variables del Sistema".

Dirección de comienzo ó nombre de la variable del sistema	*Posición de la variable del sistema*	*Contenido en memoria*
16384	—	"Archivo de información gráfica"
22528	—	"Cualidades"
23296	—	"Memoria intermedia" de la impresora
23552	—	"Variables del sistema"
23734	—	"Información microunidad de discos"
CHANS	23631	"Información de Canales"
PROG	23635	"Programa en Basic"
VARS	23627	"Variables"
E_LINE	23641	"Comando/línea que está siendo editada"
WORKSP	23649	"Datos que están siendo introducidos (INPUT)"
STKBOT	23651	"Area de conservación de registros del calculador"
STKEND	23653	"Sin usar (De repuesto)"
sp	—	"Area de conservación de registros de máquina y de sentencias GOSUB"
RAMTOP	23730	"Rutinas en código máquina del usuario"
UDG	23675	"Gráficos definidos por el usuario"
P—AMT	23732	"Final de la RAM"

Tabla 2.1 *Mapa de la memoria. El "puntero del apilamiento", sp no está alojado en la RAM y sí en el registro sp del microprocesador Z80A.*

Nos vamos a referir siempre a cada variable del sistema por la dirección en la cual está, en vez de darles un nombre a cada una de ellas, lo cual sería difícil de manejar —Por ejemplo, "PROG" es el nombre del área del programa en Basic, es decir, es la dirección de la posición de memoria que contiene información acerca del programa escrito en Basic. Los nombres se dan sólo para conveniencia del usuario o como una guía para él, pero no son reconocidos por el Spectrum. Luego si introducimos una línea que ponga:

PRINT PROG.

nos dará como resultado un mensaje de error del tipo "2 ésta variable no se encuentra" (2 Variable not found). Este mensaje aparecerá en pantalla, a no ser que la varible en Basic nombrada como PROG ha sido generada coincidentemente por el programa o por el usuario. El valor de tal variable Basic no tendría nada que ver con el valor del sistema "PROG".

PEEK y POKE(3)

(Extraer e Introducir)

El mapa de la memoria es la clave para comprender el uso que hace el Spectrum de la RAM, pero a su vez la llace para explorar la RAM son las sentencias Basic PEEK y POKE (Extraer e Introducir) que permiten al usuario leer y alterar (escribir) respectivamente el contenido de cada posición de memoria. PEEK es una función cuyo formato es:

PEEK dirección

es decir, la introducción de una función PEEK debe ir seguida de una dirección para que tenga sentido, ya que la operación va a ser la de extraer un contenido de una dirección.

Esta dirección debe ser un número entero y positivo tomado entre 0 y 65525 (para el modelo de Spectrum que tenga ampliación de memoria). o una expresión aritmética cuyo resultado, una vez evaluado, nos dé el número entero y positivo que se requiere. Es muy importante encerrar dicha operación aritmética entre paréntesis porque:

PEEK 16384 + 2

va a ser interpretada por el computador como la suma del número 2 con el resultado de extraer el contenido que hubiere en la posición 16384.

Pero si decimos:

PEEK (16384 + 2)

lo interpretará como si fuese PEEK (16386), que significa extraer el contenido de la posición de memoria dada por la dirección 16386 y este contenido va a venir dado en forma de un número entero y positivo comprendido entre 0 y 255 inclusive.

Esto fue explicado anteriormente cuando hablamos de la variable del sistema llamado "PROG" que se alojaba en la posición dada por la dirección 23635. Pero el valor de "PROG", aun siendo una dirección en la RAM, es siempre mayor que 255. Por lo tanto, necesitaremos dos direcciones adyacentes y consecutivas, cuales son 23635 y 23636, para expresarlo.

El valor de "PROG" lo podremos ver en pantalla si introducimos la siguiente línea de sentencias Basic:

PRINT "PROG = "; PEEK 23635 + 256*PEEK 23636

De este modo cualquier dirección ha de ser alojada en dos posiciones (dadas éstas por sendas direcciónes) de memoria adyacentes y que pueden ser inspeccionadas mediante:

PRINT PEEK *primera dirección deseada* + 256*peek *segunda dirección (consecutiva de la primera)*

Por ejemplo si el Spectrum trabaja son la microunidad de discos conectada, el área reservada para el "mapa" de la microunidad no debe de existir y, por

---

(3) N. del T.: Siempre que nos refiramos a sentencias en Basic, las dejaremos en su idioma original aunque entre paréntesis se dé una aclaración sobre su significado. La razón de ello es que el ZX Spectrum posee un Basic diseñado en ese idioma.

lo tanto, el área de "información de canales" deberá seguir inmediatamente al área de "variables del sistema". Así el valor de la "variable del sistema CHANS" se mantiene en la 23631 y 23632. De modo que introduciendo:

PRINT PEEK 23631 + 256*PEEK 23632

nos dará como resultado 23734.

La función PEEK puede ser usada para extraer o leer el contenido de cualquier posición en memoria, incluyendo las de contenido fijo que están en la ROM y que expresan instrucciones.

Esta función es una herramienta muy importante. El hecho de extraer el contenido de cualquier posición no va a afectar ni alterar el programa o a las variables del Spectrum. Muy ocasionalmente los resultados de una operación con PEEK pueden ser engañosos debido a que el contenido de una posición que está siendo leída puede alterarse durante o inmediatamente después de la ejecución de una instrucción. Por ejemplo, si el contenido de las posiciones asignadas a la información alojada en el ángulo superior izquierdo de la pantalla son leídas y queremos que su resultado se refleje en esa misma posición dentro de la pantalla, la información podría haber cambiado ya debido al tiempo empleado por el usuario en visualizarla.

Muy distinto es el caso del comando u orden POKE (Introducir, Escribir), ya que su uso va a resultar más peligroso que el de la función PEEK porque el comando POKE interfiere el funcionamiento del Spectrum. Así, es muy posible hacer que la información en la RAM carezca de sentido si usamos este comando y forzamos a la máquina a detener la ejecución (Halt) o abortar el programa en curso y mostrar en la pantañña algún código asignado a los diferentes tipos de errores.

El formato de este comando es:

POKE *dirección, número*

Una vez más insistimos en que la dirección ha de ser un número entero y positivo tomado entre 0 y 65535 inclusive, o una expresión aritmética cuyo resultado sea entero y positivo. En este último caso hay que hacer notar que no es esencial encerrar dicha operación aritmética entre paréntesis porque POKE es un comando y no una función, de modo que no puede ser evaluada como un entero.

El número introducido con el comando POKE en una posición dada, deberá estár entre 0 y 255 inclusive.

El Spectrum aceptará y ejecutará una orden de POKE para introducir o escribir un número en una dirección de la ROM (comprendida entre 0 y 16383), pero dicho número nunca alcanzará su destino. Este hecho puede comprobarse "poniendo en marcha" (Running) el siguiente programa:

```
10 PRINT PEEK 0
20 POKE 0,92
30 PRINT PEEK 0
```

Las líneas 10 y 30 deberán, cada una, imprimir en pantalla el valor 243 porque ocurres que es el valor contenido en la posición 0 y es fijo e inamovible (ROM).

La línea 20 no tiene efecto.

El Archivo de Información Gráfica (Display File)

La información gráfica de pantalla consiste de 24 líneas conteniendo cada una 32 caracteres. Hemos visto que el archivo de información gráfica ocupa las posiciones 16384 hasta la 22527, es decir, 6144 posiciones en total. Luego el número de posiciones usado por cada carácter es de:

6144/(24*32) = 8

El camino más fácil para obtener una visión de conjunto de cómo la información gráfica está organizada es imprimir una imagen en pantalla, "guardar" dicha información en una cinta, "limpiar" la pantalla y "cargar" la imagen de nuevo. El programa P2.1 ejecuta las operaciones descritas anteriormente, es decir, "guardar" (Save) y "cargar" (Load) el contenido de la pantalla usando el carácter gráfico incluído en la tecla asignada al número "5" para crear la imagen original.

Cuando dicha imagen es recargada desde la cinta, está claro que la información gráfica en pantalla está dividida en tres secciones, cada una de ellas compuesta de líneas de ocho caracteres.

```
100 FOR i = 0 TO 703
110 PRINT "▌" ;
120 NEXT i
130 SAVE "Imagen" SCREEN$
140 CLS
150 INPUT "Rebobinar la cinta, ponerla en marcha y presionar una
        tecla"; Z$
160 LOAD "Imagen" SCREEN$
```

Programa P2.1 *Ejemplo de programa para GUARDAR, LIMPIAR y CARGAR la imagen*

Cada línea de caracteres está dividida a su vez en ocho líneas de "elementos de imagen" (Pixels). Pero sorprendentemente el Spectrum np "carga" las ocho líneas de elementos de "imagen" que forman la primera línea de caracteres, seguidas de las ocho líneas de "elementos de imagen" que forman la segunda línea de caracteres, etc. En lugar de esto, el Spectrum "carga" las línewas de "elementos de imagen" situadas en la parte superior de las ocho primeras líneas de carácter seguidas de la línea de "elementos de imagen" siguientes a la línea superior anterior (todo ello se ha de entender que sucede dentro de la primera línea de pantalla o renglón para semejarlo con la impresión en papel) (4).

Otro camino para comprender la forma en que se construye la información gráfica es considerar en dónde se usan los ocho octetos (Bytes) que forman un carácter, que en nuestro ejemplo será el situado en el ángulo superior izquierdo de la pantalla. El primer octeto de encarga de formar el trazo superior del carácter y está localizado al comienzo del "archivo de información gráfica" en la dirección 16384. Hagamos una pausa para experimentar con la siguiente sentencia:

POKE 16384,0

Esta sentencia hace que el trazo superior de un carácter que contiene a su vez ocho elementos de imagen, desaparezca de la pantalla, es decir, queda en blanco. Por el contrario si ponemos:

POKE 16384,225

hará que todos los elementos de imagen se activen.

Introduciendo números (mediante POKE) entre 0 y 255 obtendremos el efecto de salpicar de puntos la raya superior del caracter.

La segunda raya, contando desde arriba, de elementos de imagen del primer carácter no se forma con el contenido de la dirección de memoria 16385 como pudiera pensarse. En su lugar el contenido de la posición 16385 va a activar los ocho elementos de imagen del trazo superior del caracter adyacente al primero. Recuérdese que hay 32 caracteres en una línea y ocho trazos por carácter, luego para acceder al segundo trazo empezando por arriba, del primer carácter habrá que ir a la posición de memoria dada por la dirección 16384 + 32*8 = 16640. Este razonamiento aplicado a los restantes seis trazos que forman el carácter de nuestro ejemplo (recuerde que era el caracter situado en el ángulo superior izquierdo de la pantalla). En resumen podemos decir que el carácter que nos ocupa estará formado por el contenido de las siguientes direcciones:

16384, 16640, 16896, 17152, 17408, 17664, 17920, 18176

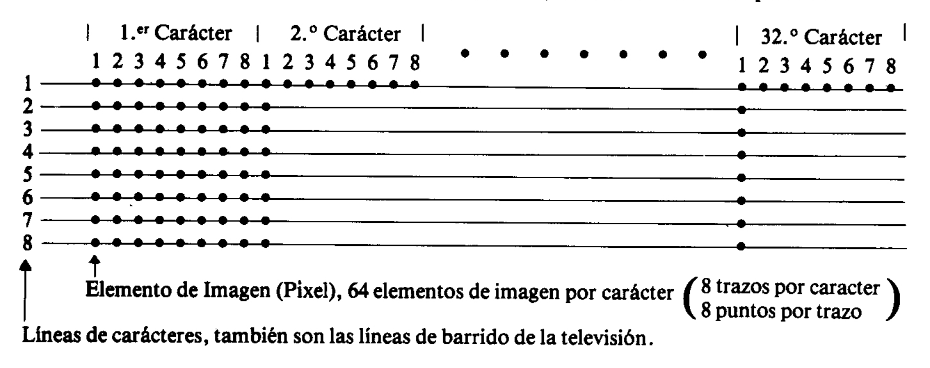
---

(4) N. del T.: La formación de un carácter en la pantalla obedece a la siguiente secuencia de hechos:

1.° El computador traza la raya superior de elementos de imagen (8 por cada carácter que determinan su "anchura") común a todos los caracteres de una línea de pantalla o renglón (32 en total por cada línea).

2.° A continuación traza la raya que sigue a la anterior (apartado 1.°) en orden descendente y que afecta a todos los caracteres de la primera línea de pantalla.

3.° Continúa trazando las sucesivas que configuran los caracteres (hasta 8 en total que es la "altura" de un carácter) de esa misma línea de pantalla.



A su vez, estas líneas son comunes para formar una línea de pantalla o renglón. Cada línea contiene 8 elementos de imagen por cada carácter a formar. Se entiende que cada línea va "montada" sobre las líneas horizontales que utiliza la televisión para formar la imagen (625 líneas, CCIR).

El programa listado en P2.2 permite al usuario experimentar por medio de la introducción (POKE) de distintos números en estas ocho direcciones de memoria. Cada posición de memoria en el archivo de información gráfica controla el estado de los ocho elementos de imagen correspondientes en la pantalla.

Este control se ejerce por medio de la conversión del número contenido en una posición dada a su forma binaria y luego exponiendo los ocho elementos de imagen de acuerdo al modelo establecido para los estados lógicos posibles de los dígitos binarios, es decir 0 ó 1. (Recuérdese nota (2) del capítulo 1.°).

```
10 REM Rutina para depositar un carácter en el ángulo superior iz-
       quierdo de la pantalla.
20 INPUT "Un carácter se forma con ocho octetos, cada uno con un
         valor entre 0 y 255 inclusive. Introducir el número del octe-
         to (entre 0 y 7)" ; n
30 IF n<0 OR n>7 OR n < > INT n THEN BEEP .2,24: GO TO 20
40 INPUT "Introducir el contenido del octeto" ; m
50 IF m < 0 OR m > 255 OR m<> INT m THEN BEEP .2,24: GO TO 40
60 POKE 16384 + 8*32*n , m
```

Programa P2.2 *Ejemplo de programa para construir un carácter en el ángulo superior izquierdo de la pantalla*

Así por ejemplo, cuando convertimos el número 240 al sistema binario convertimos el número 240 al sistema binario obtenemos:

11110000 $(1\times 2^7+1\times 2^6+1\times 2^5+1\times 2^4+0\times 2^3+0\times 2^2+0\times 2^1+0\times 2^0)$.

De manera que si en una posición tenemos el valor 240, cuatro de los ocho elementos de imagen estarán activados y cuatro no (en blanco).

Como resumen diremos que el "archivo de información gráfica" consiste en 6144 direcciones, ocho de las cuales estarán asignadas para cada posición del carácter. Cada dirección da las condiciones o el estado en que han de estar los ocho elementos de imagen de un trazo horizontal. Las direcciones asignadas a la posición de un carácter dado no ocurren de modo adyacente una de otra, sino que la información gráfica está dividida en ocho secciones y dentro de cada sección hay 256 direcciones que separan el octeto que constituye cada posición.

Las Cualidades (The Attributes)

El contenido del "archivo de información gráfica" solo determina qué elementos de imagen están activados o no en la pantalla. El "color del fondo" (semejante al color del papel sobre el que se escribe) (Paper) y la "tinta" (Ink) que se usa, así como el "brillo" (Bright) y el "destellar intermitente" (Flash) son condiciones que van a estar determinadas por el contenido del archivo de "cualidades". El área de memoria reservado a éste comprende desde la dirección 22528 a la 23295. Las "cualidades" pueden asignarse a cada una de las 768 posiciones posibles, dentro de la pantalla, correspondientes a otros tantos

caracteres. En contraste con el "archivo de información gráfica", las direcciones son asignadas a la posición que cada carácter ocupa, i. e: comenzando por el ángulo superior izquierdo, trabajando de izquierda a derecha y de arriba a abajo.

Cada dirección determina ambas cualidades, a saber: "tinta" (Ink) y "color de fondo" (Paper). Estas son cualidades determinadas por el contenido de una dirección y que afectan a la posición asignada al carácter. Los ocho colores posibles se seleccionan por medio de las teclas situadas en la fila superior del teclado del Spectrum. También determina qué posición de carácter debe de estar brillante y dónde debe destellar (Flash). Estos cuatro parámetros se codificarán de acuerdo con el cálculo siguiente:

Valor de la "cualidad" = 128*FLASH + 64*BRIGHT + 8*PAPER + INK

(Obsérvese que "FLASH, BRIGHT, PAPER e INK" son palabras inherentes al teclado Basic del Spectrum, cada cualidad está en una tecla)(5).

El "destello" (Flash) y el "Brillo" (Bright) tomarán el valor 1 cuando deseemos que esta cualidad aparezca, de lo contrario será el valor 0 el que ha de sustituirse en la ecuación. Respecto del "color de fondo" (Paper) y de la "tinta" (Ink) estos tomarán el valor del color deseado y que está grabado en el teclado (por ejemplo, si queremos rojo, pulsar la tecla correspondiente al número 2).

El programa P2.3 decodifica la cualidad, es decir que dándole un valor, éste nos mostrará en pantalla las correspondientes cualidades de "PAPER", "INK", etc.

```
 10 REM Decodificador de cualidades
 20 DATA "Black ", "Blue ", "Red  ", "Magenta", "Green ",
         "Cyan  ", "Yellow", "White ", "Bright ", "Flash"
 30 DIM c $ (8,7)
 40 FOR i = 1 TO 8
 50 READ c $ (i)
 60 NEXT i
100 REM Decodificador de cualidad
110 INPUT "Introducir un número entre 0 y 255. Este programa de-
         codifica su interpretación en el archivo de cualidades" ; n
120 IF n < 0 OR n > 255 OR n<>INT n THEN BEEP .2,24: GO TO 110
200 PRINT "El color de la tinta es" ; c $ (1 + n—8*INT(n/8))
210 PRINT "El color de fondo es" ; c $ (1 + INT(n/8)—8*INT(n/64))
220 IF INT (n/64) = 1 OR INT (n/64) = 3 THEN PRINT "El carácter
    está brillante"
```

(5) N. del T.: Hay que tener en cuenta que cuando queramos establecer las cualidades de "FLASH" y "BRIGHT" fuera de un programa escrito en Basic, como por ejemplo en código máquina, éstas no se activan con las teclas marcadas con tales anagramas. Solo se activarán dándole el valor lógico del estado deseado, y para activarlo o desactivarlo. El resto de la ecuación se establece de acuerdo al párrafo que va después de la igualdad.

```
230 IF n > 127 THEN PRINT "El carácter deberá destellar"
300 PRINT AT 6,0 ; "■■■■■■■■■■■■■■■■■■■■■■■■■■
■■■■■■"
310 FOR i = 22720 TO 22751
320 POKE i , n
330 NEXT i
500 INPUT "Si quiere repetir pulse ENTER" ; z $
510 CLS
520 GO TO 110
```

Programa P2.3 *Ejemplo de programa para decodificar una cualidad*

La "Memoria Intermedia" de la impresora[6] (The Printer Buffer)

Las 256 direcciones de RAM siguientes al área de cualidades se usan para almacenar temporalmente una línea de caracteres incompleta (en el sentido de que ocupa más de 32 caracteres que caben en una línea de caracteres o renglón). Esta línea incompleta se transferirá posteriormente a una impresora. Este área es necesaria porque un programa en Basic puede ordenar imprimir una parte de una línea terminada por un punto y coma o por una coma, para indicar que el resto de la línea no ha llegado todavía. En ciertas circunstancias el comando "TAB" (Tabular) puede actuar de manera semajante. Esa parte de la línea no es enviada inmediatamente a la impresora, debido a que ésta puede dar salida sólamente a una línea completa y para continuar necesita un tiempo para avanzar el papel y prepararse para la siguiente. Por consiguiente, esa parte de la línea se almacena temporalmente en el "área de memoria intermedia" reservada a la impresora hasta que el programa le comunique el resto de la línea.

Muchas de las rutinas de la sección "B" hacen uso de este área para enviar datos desde el programa Basic o desde el teclado a las rutinas. El área de "memoria intermedia" es conveniente para este propósito debido a que su dirección es fija y es improbable que el usuario desee usarla para otro fin cuando llame a una rutina en código máquina.

Area reservada al programa en Basic

Si una microunidad de discos está conectada al Spectrum, el comienzo del área reservada al programa en Basic va a estar determinada por la variable del sistema llamada "PROG" localizada en la dirección 23635. En ausencia de

---

(6) N. del T.: También llamada"memoria tampón". Siempre que hablemos de un periférico asociado a un computador (en este caso la impresora), debemos tener en cuenta su enorme lentitud de trabajo, en comparación con la velocidad de las operaciones internas que lleva a cabo el computador. Debido a esta diferencia de velocidades de trabajo es por lo que se hace necesario disponer de una memoria intermedia de capacidad reducida que haga posible recibir y transmitir (en el caso de la impresora sólo transmitir pues ésta sólo recibe información) datos a regímenes de velocidad diferentes (Buffer).

dicha microunidad de discos, el área comenzará en la dirección 23755. De ahora en adelante asumiremos que la microunidad no esfa conectada.

El programa listado en P2.4 imprime el contenido de las primeras 18 direcciones de este área reservada al programa (en Basic) como se muestra en la F2.1. Estas 18 direcciones se usan para almacenar la primera línea del programa dado:

10 REM Peek program

Se puede aprender mucho acerca del método de codificación de programas estudiando con detenimiento el listado dado en F2.1.

```
10 REM Peek program
20 FOR i = 23755 TO 23772
30 PRINT i , PEEK i
40 NEXT i
```

Programa P2.4 *Ejemplo de programa para IMPRIMIR el contenido de las primeras 18 direcciones en el área de programa*

23755	0	
23756	10	
23757	14	p
23758	0	e
23759	234	e
23760	80	k
23761	101	
23762	101	p
23763	107	o
23764	32	
23765	112	p
23766	114	r
23767	111	o
23768	103	g
23769	114	r
23770	97	a
23771	109	m
23772	13	

Figura F2.1 *Forma en la que se almacena dentro del área de programa, la línea 10 REM. Programa para extraer (realmente el código ASCII resultante permanece a la línea 10 REM PEEK Programa para extraer)*

El número de línea, 10, está almacenada en las dós primeras posiciones en la forma siguiente:

número de línea = 256*PEEK *primera dirección* + PEEK *segunda dirección*

Adviertase que la convención usada por el microprocesador Z80A, que consiste en multiplicar el contenido de la segunda dirección por 256 y luego sumarle el contenido de la primera dirección, no se aplica en este caso.

Dicha convención se aplica a las dos direcciones siguientes, 23757 y 23758. Estas dos direcciones juntas almacenan la longitud que va a tener el resto de la línea comenzando en la dirección 23759. El número que será almacenado en este caso es el:

14 + 256*0 = 14

De esta manera la siguiente línea comenzará su almacenamiento en la dirección:

23759 + 14 = 23773

La dirección 23759 almacena automáticamente el número 234, que es el "número clave" (Character Code) asignado a la sentencia en Basic REM(7).

Las doce direcciones siguientes contienen los números de código correspondiente a las once letras así como el espacio entre ellas el título "Peek program".

Finalmente, la dirección 23772 contiene un 13 que es el código asignado a la tecla "ENTER" indicando que ese es el final de la línea. La tabla 2.2. resume el método de codificar programas en el área reservado a los programas en Basic.

Direcciones	*Contenidos*
1.ª y 2.ª	Número de la línea almacenado en orden contrario a la convención establecida para el Z80A.
3.ª y 4.ª	Longitud de la línea excluyendo las primeras cuatro posiciones.
5.ª	El código de comando
Ultima	El carácter "ENTER", que es el número 13.

Tabla 2.2 *Método usado para codificar las líneas de un programa.*

Un punto que ha sido omitido en la tabla es la descripción del método usado para almacenar valores que tienen lugar en un programa. El método puede ser explorado por medio de la sustitución de la línea 10 por:

10 LET a = 1443

en el programa P2.4. La figura 2.2 nos muestra el resultado de ejecutar el programa con ésta nueva línea.

23755	0
23756	10
23757	14
23758	0
23759	241
23760	97
23761	61

(7) N. del T.: "REM" es abreviatura de "REMARK", que significa COMENTARIO. No indica acción específica al Spectrum, solo reserva espacio.

23762	49
23763	52
23764	52
23755	51
23766	14
23767	0
23768	0
23769	163
23770	5
23771	0
23772	13

Figura 2.2 Ejemplo de la forma en que se almacena la línea 10 en el área reservada al programa

Vemos que las direcciones 23755 hasta la 23758 siguen como antes. Les siguen las que contienen los códigos asignados a LET, a, = ,y los cuatro dígitos que juntos forman el número 1443. El contenido de la dirección siguiente, esto es el de la 23766 es 14, éste resulta ser el "número de código" que indica que las cinco direcciones siguientes contienen el número 1443 en forma numérica. La línea termina en la dirección 23772 cuyo contenido es el código de la tecla "ENTER", ed decir 13.

Forma numérica en 5 octetos

Se usan cinco direcciones para almacenar cada número que aparezca en un programa Basic (Excepción hecha para el caso en que el número exprese la secuencia de líneas de un programa, como hemos visto ya). Números enteros entre —65535 y +65535 son almacenados de manera semejante al convenio usado por el microprocesador Z80A. Para el caso de estos números las primeras direcciones y la última contienen un cero cada una y la tercera y la cuarta contienen al número en la forma:

número = PEEK *tercera dirección* + 256*PEEK *cuarta dirección*

Veamos un ejemplo.

Sea el número 16553, éste va estar contenido en cinco direcciones de la forma siguiente:

0 0 169 64 0

ya que:

169 + 256*64 = 16553

Los números no enteros son tratados en forma de "coma flotante" (Floating Point) es decir como un exponente contenido en la primera dirección y una mantisa almacenada en las cuatro restantes, por ejemplo:

número = mantisa * 2 ↑ exponente

La primera direción donde se aloja la mantisa se usa también para determinar el signo del número. Si la dirección contiene un valor entre 0 y 127 el número es positivo, en caso contrario es negativo.

El programa P2.5 se puede usar para reconstruir un número no entero a partir del valor dado a sus cinco componentes.

```
10 PRINT"Enter the exponent and the four bytes of the mantisa. All
entries to lie between 0 and 255 inclusive"
20 INPUT e, a, b, c, d
30 PRINT , , "Exponent = " ; e
40 PRINT "Mantissa = " , a , , b , , c , , d
50 PRINT, , "The number = " ; (2*(a< 128)—1)*2↑ )(e-160)*(((256*
(a + 128*(a < 128)) + b)*256+d)
```

Programa P2.5 *Este programa reconstruye un número no entero a partir del valor dado a sus cinco componentes*

El Area de Variables

El área de variables comienza en la dirección contenida en la "variable del sistema" (VARS) contenida a su vez en la dirección 23627. Cuando se declara una nueva variable, ya sea por programa o desde el teclado, se genera una cierta cantidad de espacio en dicha área.

Todos los nombres de variables han de comenzar con una letra y no se hace distinción entre mayúsculas y minúsculas. Estas restricciones permiten al Spectrum manipular el código de la letra inicial de cada variable, de modo que pueda distinguir entre los seis tipos de variables permitidas, con solo inspeccionar el rango, dentro del cual está cada código. Todas las variables numéricas con un nombre encabezado por una letra tienen códigos asignados dentro del rango que va desde el código 97 al 122; la letra a es el 97; la letra b es el 98; la c es el 99, etc. De manera similar, las matrices numéricas tienen códigos que oscilan en el rango del 129 al 153; a es 129; b es 130; c es 131, etc. Los rangos de los códigos se resumen en la tabla 2.3. La longitud de cada tipo de variable se muestra en la tabla 2.3.

Tipo de variable	*Rango del código*	*Longitud en el área de variable*
Numérica (cuyo nombre tenga un sólo caracter	del 97 al 122	6
Numérica (con nombre de varios caracteres)	de 161 al 186	5 + longitud del nombre
Matriz numérica	del 129 al 154	4 + 2* número de dimensión + + 5*número total de elementos
Variable de control de un lazo FOR-NEXT	del 255 al 258	18
De "cadena" (STRING)	del 65 al 90	3 + longitud de la cadena
Matriz de caracteres	del 193 al 218	4 + 2*número de dimensión + número total de elementos

Tabla 2.3 *Variables, rango de sus códigos y longitudes*

Rutinas en ROM

Algunas rutinas de la sección "B" usan rutinas contenidas en la ROM del modo siguiente:

rst 16

Imprime (PRINT) el contenido del registro acumulador.

Call 3976

Inserta el caracter contenido en el acumulados en la dirección de la RAM dada por el registro dual hl (o pareja de registros hl).

Call 4210

Borra el carácter contenido en la dirección de la RAM dada por el registro dual hl.

Call 6326

Si el acumulador contiene el número 14, activa el "indicador de cero" (Zero Flag) e incrementa el contenido del registro dual cinco veces.

Call 6510

Devuelve en el registro dual hl la dirección en la RAM de la línea cuyo número de secuencia fue entregado a la rutina en el registro dual hl.

3. LENGUAJE MAQUINA DEL Z80A

Da comienzo este capítulo con la explicación de algunas de las palabras más importantes que se van a usar, tales como: "BIT" (BIT)(1), "OCTETO" (BYTE), "DIRECCIÓN" (ADDRESS) y "REGISTRO" (REGISTER).

Estas palabras serán usadas continuamente a lo largo del libro. El número y variedad de registros del Z80A será examinado después, con especial hincapie en lo que se refiere a un pequeño número de instrucciones. Finalmente haremos un pequeño resumen del "juego de instrucciones" características del Z80A (Instruction Set).

Quizas el aspecto más difícil, para el no iniciado en la programación en código máquina, sea la asimilación del gran número de palabras y conceptos.

Sin embargo, antes de entrar de lleno en la parte principal del capítulo vamos a examinar una instrucción, como un ejemplo de lo que vendrá más adelante. Vamos a considerar una instrucción compuesta, con la que nos vamos a encontrar en muchas de las rutinas de la sección "B":

ld hl (23627)

La instrucción la podemos leer como:

"Carga" el "registro" dual hl con los "octetos" contenidos en las "direcciones" 23627 y 23628.

La instrucción es convertida en forma de tres números decimales: 42, 75, 92. El primer número significa:

ld hl, (   )

o lo que es igual: "Carga" el "registro" dual hl con el contenido de dos "direcciones" consecutivas de memoria. Las direcciones en cuestión están especificadas por el segundo y tercer número, mediante el cálculo siguiente:

dirección más baja = primer número + 256*segundo número
dirección más alta = dirección más baja + 1

Que aplicado a nuestro caso, será:

dirección más baja = 75 + 256*92 = 23627
dirección más alta = 23627 + 1 = 23628

La palabra "carga" (Load) es solamente otra forma de decir "copia" o "transfiere" y en cuanto a *h* y *l* pueden ser tomados como dos posiciones muy concretas dentro del microprocesador Z80A, que tienen como fin el almacenamiento de números. La instrucción completa significará: "copia" el contenido de 23627 en el registro *l* y el contenido de la dirección 23628 en el registro *h*.

Adviértase que la dirección maś baja va a ser la "fuente" para el registro *l* y la dirección más alta será la "fuente" para el registro *h*.

---

(1) N. del T.: "BIT" es la contracción obtenida a partir de las palabras inglesas "**B**inary Dig**it**". Vamos a optar por expresarla así siempre ya que en español no existe una traducción que utilice una sola palabra para expresarla. Diremos que "BIT" es la unidad elemental de información que puede adoptar dos valores o estados lógicos distintos, 1 ó 0.

BIT

Un *bit* es la unidad fundamental de memoria de cualquier computador, porque un bit solo puede existir en uno de los dos estados posibles. Los dos estados pueden ser expresados como ENCENDIDO o APAGADO, ABIERTO o CERRADO, CIERTO o FALSO, SI o NO, ARRIBA o ABAJO, MACHO o HEMBRA, o cualquier otro par de condiciones lógicas opuestas. El mecanismo por el cual la memoria de un computador trabaja no es realmente importante para nosotros. A Grandes rasgos podemos decir que el Spectrum memoriza el estado de un bit por medio de la posición abierta o cerrada de un microscópico interruptor de estado sólido.

Diremos que la notación usual es que un estado es el estado 0 (cero) y que el opuesto es el estado 1.

Un bit se considera que está "activado" (Set) cuando se está en estado 1 y estará "desactivado" (Reset) cuando esté en estado 0 (cero). Esta notación nos permite hablar de una cadena de "bits" en términos de su equivalente binario y por la conversión de un número binario decimal. De modo que cada cadena de bits se pueda expresar por medio de un número único, entero y positivo.

Consideremos, por ejemplo, una cadena de ocho bits en la cual los cuatro bits más a la derecha están activados (1) y los cuatro más a la izquierda están desactivados (0). Este modelo de cadena de bits se ilustra en la tabla 3.1.

Interruptor	Apag.	Apag.	Apag.	Apag.	Encend.	Encend.	Encend.	Encend.
Posición	abierto	abierto	abierto	abierto	cerrado	cerrado	cerrado	cerrado
Cadena binaria	0	0	0	0	1	1	1	1
Número del bit	7	6	5	4	3	2	10	

Tabla 3.1 *Un grupo de 8 bits con los cuatro bits más a la izquierda desactivados y los cuatro más a la derecha activados*

Una cadena binaria puede convertirse a su forma decimal si se recuerda que en un número binario, la columna situada más a la derecha expresa la unidad, la siguiente a la izquierda expresa el valor 2, la siguiente a la izquierda toma el valor 4, etc. Es decir, hay que tomar el valor doble en cada movimiento a la izquierda. El número decimal equivalente de la cadena binaria expresada en la tabla 3.1 será:

$$00001111 = 0*128 + 0*64 + 0*32 + 0*16 + 1*8 + 1*4 + 1*2 + 1*1 = 15$$

Debido a que hay un 1 como coeficiente de las columas cuyo peso es 1, 2, 4 y 8 y el resto de los coficientes son cero aunque las columnas tengan pesos de 16, 32, 64 y 128.

Referirse a los bits por su situación geográfica (el más a la derecha, el segundo desde la izquierda, etc.) no deja de ser un inconveniente. De modo que adoptaremos el convenio de numerarlos de derecha a izquierda empezando por el número O. Aunque no es enteramente casual, cuando usamos este

convenio, el número del bit, es también el valor del exponente al que hay que elevar el número 2 (base binaria). Por ejemplo:

$2^{\text{número del bit}}$ = valor de columna

Vemos que en efecto, el bit número 3, por ejemplo, aparece en la columna de peso 8.

Octetos (Bytes)

El microprocesador Z80A que integra el corazón del ZX Spectrum opera con grupos de 8 bits a la vez (el término "opera" cubre las diferentes tareas que están implementadas en el "juego de instrucciones" (Instruction set) tales como: suma, resta, rotación, operador lógico "Y" (And), etc.). La forma de estas instrucciones se explicará más adelante dentro de este capítulo.

Si bien un bit es la unidad fundamental de la memoria de un computador, los bits se manipulan generalmente en grupos de ocho. Estos grupos de ocho bits son llamados "Octetos" (Bytes).

Se pueden usar cada uno de estos octetos en una RAM, para albergar un simple número entero y positivo comprendido entre 0 y 255 onclusive, activando o desactivando los ocho bits que componen un octeto de acuerdo con el equivalente binario del número. El octeto de la tabla 3.1, por ejemplo, contiene el número decimal 15.

Hay 16384 en la "Memoria de Sólo Lectura" (ROM) que posee el ZX Spectrum y el contenido de estos octetos, junto con la organización electrónica, es lo que le da al computador sus características. Los contenidos se introducen en la ROM en el momento de su fabricación y, por consiguiente, no pueden alterarse. Esta es la razón de su nombre: "Memoria de Solo Lectura" (ROM). El contenido de esta memoria ROM solo se puede leer, pero no borrar, ni cambiar por otro.

La versión del Spectrum que no lleva ampliación de memoria contiene otros 16384 octetos de "memoria de acceso aleatorio" (RAM). El término "acceso aleatorio" es de alguna forma una designación errónea. Esto no quiere decir que la memoria se use al azar, lo que realmente significa es que se puede acceder a cualquier octeto (Byte) de modo inmediato y en cualquier momento. Esta propiedad contrasta con una memoria de acceso secuencial, como puede ser una cinta de cassette, en la que es necesario bobinar o rebobinar hasta llegar a la porción requerida, o leer todos los datos de la cinta anteriores al que realmente queremos.

Para el no iniciado diremos que 16384 no parece ser un número de octetos cómodo de manejar. Sin embargo, de hecho es un número apropiado ya que 2^{14} = 16384 (16384 es igual a 2 multiplicado por si mismo 14 veces).

Refiriéndonos al mundo de los computadores, las potencias de 2 son maneras de "redondear" números al igual que las potencias de diez —como son los cientos (10^2), miles (10^3), etc.— que también son maneras de redondear números y que usamos todos los días. Un caso particular de redondear números es 1024, o lo que es igual 2^{10}. El número 1024 está lo suficientemente próximo a 1000 como para justificar el uso de la letra K (kilo) para represen-

tarlo. Luego, 1024 se escribirá como 1 K y para el caso del número16384 (que es 16×1024), lo escribiremos como 16 K.

Direcciones (Addresses)

Un computador debe ser capaz de identificar cada posición dentro de su memoria, de manera que pueda "transferir" información a y desde una posición concreta. Desde luego, cada posición viene dada o determinada por una dirección única. Una dirección es un número entero y positivo mayor o igual a cero.

Muchas de las instrucciones del Z80A son de la forma: "Copia el contenido de las siguientes direcciones en tal y cual registro o pareja de registros (Registro dual)". Por ejemplo la instrucción:

ld hl (23627)

que fue descrita al principio de este capítulo, obedece a esta forma. La dirección que sigue a la instrucción se alberga en dos octetos y así, el número de posiciones a las que puede acceder el procesador estarán limitadas únicamente por el número de direcciones que se pueden expresar por medio de dos octetos. Este número será el mismo que los posibles números que se puedan contener con cadenas o grupos de 16 bits, es decir $2^{16}=65536$. Recuerde que una dirección (dada por dos octetos) se interpreta de la forma:

dirección = primer octeto + 256*segundo octeto

Algunas veces los octetos se llaman octeto bajo y octeto alto o si se prefiere, inferior y superior, respectivamente.

El formato, en dos octetos, que obtendremos para la dirección 16384 (que es la dirección de comienzo de la RAM en el Spectrum) es, octeto bajo = 0; octeto alto = 64. Porque:

0 + 256*64 = 16384

Los Registros del Z80A

Un computador no sabe alterar el contenido de la memoria directamente cuando ejecuta un programa. En lugar de ello, copia el contenido de una posición de memoria en un registro y opera sobre el contenido de ese registro. Los registros tienen una función similar, en lenguaje de máquina, a las variables en Basic que se usan para almacenar números y se emplean para controlar una decisión. Sin embargo, difieren de las variables en Basic en que los registros están limitados en número y existen en el interior del microprocesador como tal y no en la RAM. También es cierto que los registros pueden albergar un solo octeto, o dos octetos en el caso de un registro dual.

El Z80A es un microprocesador de mucha potencia en cuanto a que posee varios registros y así puede dar cabida a varios números a la vez, reduciéndo la necesidad de hacer transferencias entre procesador y memoria que conllevan bastante consumo de tiempo.

La mayoría de estos registros tienen una o más propiedades especiales.

El Registro Acumulador "a"

El acumulador es el registro más importante, porque muchas de las instrucciones aritméticas, suma por ejemplo, y las instrucciones lógicas, como el "O" Lógico (Logical "OR") operan sobre el contenido de ese registro.(2)

De hecho éste toma ese nombre porque acumula en el registro "a" el resultado de varias operaciones sucesivas.

Algunas de las instrucciones que se refieren al acumulador usan un segundo registro o una dirección de memoria como "fuente" de datos. Por ejemplo, add a,b (suma a,b) da instrucciones al procesador para sumar el contenido del registro b al contenido del registro "a", depositado el resultado en "a".

El "Registro Indicador" f (Flag Register)

Muchos de los registros se toman por parejas, en el sentido de que algunas instrucciones operan con dos registros a la vez. El registro indicador f es dual con el registro a, aunque en este caso el enlace que los une es débil, porque está limitado a las instrucciones del tipo "empujar" (o "meter") (Push), "hacer saltar" (o "sacar") (Pop) e "intercambiar" (Exchange).

El registro *f* es bastante diferente del resto debido a que los ocho bits que lo componen se usan a modo de indicadores para tener constancia y controlar la secuencia de ejecución de un programa. Cada "indicador" (Flag) se usa para señalar que uno de los dos estados lógicos posibles (0 ó 1) ha tenido lugar. Por ejemplo, el "indicador cero" (Zero Flag) nos dice que el resultado de la última operación aritmética ejecutada (suma, resta, etc.) fue cero. Para la mayoría de los usuarios sólo cuatro de los ocho indicadores (uno por cada bit del registro *f*) tienen interés. Sus propiedades se resumen en la tabla 3.2.

El "indicador signo" (Sign) es el más simple. Por convenio, si se está usando un octeto que representa un número afectado de signo, el bit número 7 se usará para retener el signo, siendo "activado" (Set) (en estado lógico 1) cuando el número es negativo y "desactivado" en el caso contrario. El indicador signo refleja el signo del último resultado obtenido.

El indicador "cero" estará activado si el resultado de la última operación fue cero. También interviene en las instrucciones de comparación que son en cierto modo instrucciones de "resta", para las que el resultado se descarta.

El "indicador de acarreo" (Carry Flag) señala el "rebosamiento" (Overflow) que ocurre si el resultado de una suma es demasiado grande para almacenarse en el registro y también en el caso en que una operación de "resta" ocurra con un número mayor en el sustraendo. Hay también algunas instruc-

---

(2) N. del T.: Hemos visto que un computador puede realizar operaciones aritméticas, pero también puede realizar operaciones lógicas, que son de importancia cuando el programa ha de tomar decisiones acerca de expresiones alfanuméricas. Dos proposiciones (o expresiones) emplazadas por el cuantificador universal (Operador Lógico) "Y" (And) dan como resultado una proposición verdadera, cuando ambas proposiciones componentes lo son a la vez el resultado será falso en los demás casos. Si a partir de dos proposiciones componentes, efectuamos con ellas la operación lógica "O", el resultado será una proposición cierta si alguna de las dos (o ambas a la vez) lo son. Será falsa solo cuando ambas proposiciones lo sean.

ciones de rotación en las que los bits del registro *a* son rotados a la izquierda o a la derecha, con los bits 7 y 0 rotando a/desde el indicador de acarreo.

Indicador	*Abreviatura*	*Abrev. cuando está desactivado*	*Empleo*
Signo	M	P	Estará activado si el último resultado fue negativo
Cero	Z	NZ	Activado si el último resultado fue cero ó una igualdad
Acarreo	C	NC	Activado cuando el último resultado es demasiado largo como para caber en en un octeto (ó en dos octetos para operaciones con registros duales).
Paridad/Rebose	PE	PO	En cuanto a paridad, estará activado cuando el resultado de la última operación tiene paridad impar/Rebose; está activado cuando una operación hace cambiar al bit 7 como resultado de un rebose de los otros bits.

Tabla 3.2 *Lista de los cuatro indicadores más interesantes para el usuario, que controlan la mayoría de las operaciones del Z80A*

El indicador de Paridad/Rebose es realmente dos indicadores en uno. Es usado como un indicador de rebose por las instrucciones aritméticas para indicar si el bit 7 ha sido alterado por un acarreo o rebose generado por el bit 6. Por lo tanto, se usará para probar si se ha alterado el bit del "indicador de signo". Las instrucciones lógicas usan el mismo indicador para indicar la paridad del resultado (la paridad en un número expresado en forma binaria es el número de bits "activados", es decir, puestos a 1). Si el número es par, se dice que su paridad es par e impar en caso contrario.

El indicador estará activado si la paridad de un resultado es impar.

El efecto de algunas instrucciones depende del emplazamiento de un indicador en particular. Por ejemplo, la instrucción:

```
jr z, d
```

causa que el procesador (Z80A) salte sobre las siguientes "d" instrucciones (*d* indica una cantidad de desplazamiento que ha de saltar) si el indicador "cero" está activado. Si el indicador "cero" no está activado, el procesador ejecuta la siguiente instrucción en secuencia, como es usual. Hemos visto que

el "registro indicador" es importante porque permite al procesador tomar decisiones y bifurcarse hacia otra parte del programa.

Los Registros Contadores *b* y *c*

El registro *b* y su ampliación al registro *c,* con el cual está emparejado, estará disponible para un cierto número de propósitos. No obstante, su uso más importante es como contador. Hemos visto ya como el flujo de un programa puede controlarse mediante el uso del indicador "cero" con la instrucción jr z, d. Otra instrucción como la dada por:

djnz d

también usa el indicador "cero', para permitir la construcción de "lazos" (Loops) en código máquina, usando *b* como contador, en analogía con los lazos definidos en Basic por medio de las sentencias del tipo "For-Next" (Para-Siguiente).

Cuando el Z80A encuentra esa instrucción, el contenido del registro *b* decrece en una unidad. Si el resultado es cero, prosigue con la ejecución de la siguiente instrucción en secuencia.

Si el resultado no es cero, la rutina salta un cierto número de instrucciones dado por "d". Si el programador usa un valor negativo para "d", el salto ocurre hacia las instrucciones más cercanas al principio del programa. Asumiendo que no hay otras bifurcaciones, el procesador podría, eventualmente, encontrar la misma instrucción. Por supuesto que cargando inicialmente el registro *b* con un valor apropiado y ajustado adecuadamente el desplazamiento "d". una sección de código máquina puede ser ejecutada un número dado de veces.

El registro *b* almacena un solo octeto y éste podrá variar entre 0 y 255 inclusive, de modo que como máximo se pueden hacer 255 pasos a través de la misma sección de código usando este procedimiento.

No existen medios similares para hacer más de 255 pases a través de un lazo, pero sí existen un número limitado de potentes instrucciones que usan todos los 16 bits del registro dual bc como un contador, lo que implica tener 65535 pases posibles:

Un ejemplo es la instrucción:

cpdr

Cuando el Z80 la encuentra, ejecuta las acciones siguientes:

1) Decrementa el registro dual bc en una unidad.
b) Decrementa el contenido del registro dual hl.
3) Compara el contenido del acumulador *a* con el contenido de la posición de memoria cuya dirección está almacenada en hl.

El procesador repite estas acciones hasta encontrar una igualdad entre el contenido de *a* y el contenido de la posición de memoria ó hasta que bc = 0. Se puede usar esta instrucción para buscar un número en particular a través de toda la memoria.

El Registro Dirección "de" y "hl"

Los registros *d* y *e* no tienen ninguna función individual, y la mayoría de las veces se usan como almacenamiento temporal, de acceso rápido a memo-

ria. Se pueden usar juntos también para almacenar la dirección de una posición de memoria que en ese momento nos interese.

La principal función de los registros *h* y *l* agrupados es almacenar la dirección de una posición en memoria y ya hemos visto como ciertas instrucciones bastante potentes hacen uso del hl para este propósito. Al octeto superior se dedica *h* y *l* se dedica al octeto inferior. La dirección se almacena en la forma:

dirección = 256*h + l

dando un máximo de 65536 direcciones únicas (del 0 al 65535 inclusive).

Los Registros Indice "ix", "iy"

Los registros *ix* e *iy* tienen 16 bits cada uno y pueden usarse solamente como tales, en contraste con los registros duales *bc, de* y *hl* que se pueden usar por parejas (16 bits) o individualmente (8 bits). Generalmente se usan los registros *ix* e *iy* de manera similar al registro dual *hl*, a pesar de que las instrucciones que los emplean requieren un octeto más para su almacenamiento, en comparación con las instrucciones equivalentes que usan el *hl*.

Sea el ejemplo:

```
add hl,bc
```

ésta es una instrucción de un octeto que hace que el Z80A sume el contenido del registro dual *hl* con el *bc* y deje el resultado en el *hl*. La misma instrucción usando *ix* e *iy* puede ponerse como:

```
add ix,bc
```

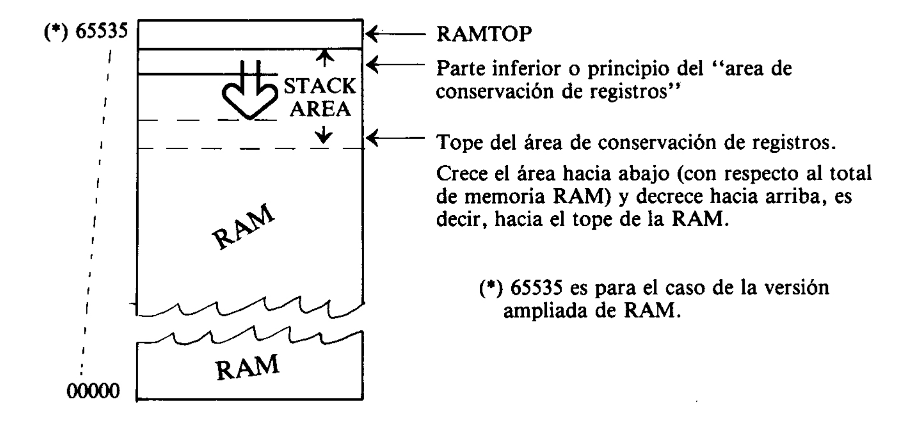
y esta sí es una instrucción de dos octetos. Los registros *ix* e *iy* tienen una propiedad más, de la que no dispone el registro dual *hl*. Esta propiedad radica en que tanto *ix* como *iy* se pueden usar con un desplazamiento "d". Esto significa que una instrucción referida como (ix + d) no usa la posición de memoria cuya dirección está contenida en ix. En lugar de ello, *d* se suma con el valor de *ix* para dar una nueva dirección, y la instrucción usa luego la dirección de memoria correspondiente. Esta es la propiedad que hace a *ix* e *iy* llamarse "registros índices".

El "puntero de apilamiento" sp

El "área de conservación de registros" (Stack) es una zona, en ó cerca de la parte alta de la RAM, que se usa para el almacenamiento temporal del contenido de los registros duales. Esta concebida de manera que crece hacia la parte inferior de la RAM según se va llenando y se encoge hacia la parte superior de la RAM según se vacía. Los párrafos que siguen aportan más luz sobre este mecanismo. La parte más baja ó principio del "area de conservación de registros" (Stack) es fija y en el ZX Spectrum, ésta se halla inmediatamente por debajo de la posición apuntada por la variable del sistema llamada "RAMTOP". La parte alta o tope del "area de conservación de registros" está por debajo del principio o parte más baja de dicho área debido a que ésta "crece hacia abajo" y "encoge hacia arriba"(3).

---

(3) N. del T.: Quizá sirva como aclaración una pequeña figura que ayude a comprender el mecanismo del Stack.



La direción de la posición que en cada momento tiene el tope o parte alta del área de conservación de registros está en el registro *sp*.

Las transferencias a/desde el área de conservación de registros se hace por medio de instrucciones del tipo "empujar" (Push) y "sacar" o "hacer saltar" (Pop)(4).

Sea el ejemplo dado por la instrucción:

push hl

ésta hace que el procesador ejecute:

1) Decrementa sp.
2) Copia el contenido de h en la posición apuntada por sp.
3) Decrementa sp.
4) Copia el contenido de l en la posición apuntada por hl.

La instrucción opuesta "hacer saltar" (Pop) ejecuta justamente la secuencia a la inversa. De esta manera, el par de valores más recientes es "empujado" al interior del área de conservación de registros y son siempre éstos mismos valores los que son hechos saltar de nuevo. Este mecanismo nos provee de un método simple y conveniente de almacenamiento temporal de registros. Por ejemplo, mientras se está llamando a una subrutina. Dando los registros duales que han de ser "sacados" o "hechos saltar" en orden inverso a como originalmente fueron "empujados" no debería haber problemas.

El contador de programa pc

El contador de programa pc (Programa Counter) es un registro de 16 bits muy importante porque contendrá siempre la dirección en memoria de la siguiente instrucción a ejecutar.

---

(4) N. del T.: En este tipo de instrucciones, el sentido que las palabras inglesas "push" y "pop" nos quieren dar es el de "empujar" (la acción es la misma que la de introducir un valor) y "hacer saltar" o "sacar" respectivamente. Aunque la acción es la de introducir un valor no es exactamente la misma ya que "empujar" conlleva el desplazar el resto del apilamiento y en el caso de "hacer saltar" o extraer un valor, es lo contrario.

El flujo normal de sucesos, cuando se ejecuta una instrucción es el siguiente:

1) Copia el contenido de la posición apuntada por el registro pc en un registro especial dentro del procesador.
2) Si la instrucción está contenida en varios octetos, incrementa el *pc* y copia el contenido de la posición siguiente en un segundo registro especial.
3) Incrementa el *pc* de manera que esté apuntando a la siguiente instrucción que va a ejecutarse.
4) Ejecuta la instrucción que acaba de ser copiada.

Una instrucción de "salto" tal como *djnz d* ó *jrz d* altera el flujo normal de secuencias o sucesos alterando el pc durante el paso reseñado en 4). Obsérvese que esta alteración ocurre después de que el pc ha sido incrementado, de manera que el valor del desplazamiento "d", se deberá calcular siempre relativo a la posición de la instrucción que sigue a la que contiene el desplazamiento.

Los "registros de intercambio" af', bc', de' y hl'

El Z80A posee duplicados de cada unos de los registros a, b, c, d, e, h y l. Los registros duplicados se ditinguen por el uso de un apóstrofe (Prime) en su identificación. Por ejemplo, a' es el duplicado del registro a. Las instrucciones no operan sobre estos duplicados directamente, pero disponemos de las "instrucciones de intercambio" para cambiar uno por otro dos o más registros fuera de uso y poner en lugar de ellos sus duplicados. Las instrucciones de intercambio se ejecutan muy rápidamente, mucho más que las de "empujar" (Push) y "sacar" (Pop), por ejemplo. Mas bien, lo que ocurre es que se cambian un conjunto de interruptores internos de manera que el registro identificado por un apóstrofe (prima) se usa por consecutivas instrucciones y el registro original permanece en suspenso.

Algo sobre el "Juego de Instrucciones Caracteristicos (Instruction Set)

Hay más de 600 elementos que componen el juego de instrucciones características del Z80A listado en el apéndice "A". Como hay solamente 256 combinaciones diferentes de 8 bits (ya que $2^8 = 256$), algo menos de la mitad de las instrucciones deberán estar contenidas en un octeto. Las restantes instrucciones están contenidas en dos o incluso tres octetos.

El primer octeto, correspondiente a una instrucción de dos octetos, deberá ser: 203, 221, 237 ó 253 (CD, DD, ED ó FD en hexadecimal). Los dos primeros octetos de una instrucción de tres octetos será 221, 203 ó 253, 205 (DD, CB ó FD, CB en hexadecimal).

Algunas instrucciones van acompañadas de un octeto que indica el desplazamiento "d", o un número de un octeto, n, o un número o dirección de dos octetos, nn, al cual se refiere la instrucción.

Así, una instrucción sencilla puede ocupar como mucho cuatro octetos en total. Por ejemplo la instrucción:

jrnz, d

que ya vimos anteriormente, requiere un octeto para almacenar la instrucción propiamente dicha (32 en decimal, 20 hexadecimal) y un segundo octeto para almacenar el desplazamiento.

En este capítulo, cuando nos refiramos a las instrucciones lo haremos mediante sus abreviaturas o nemónicos, correspondientes al "lenguaje de ensamblamiento" ("Assembly Language") ó "código de operación" (Op code). Los nemónicos son una manera abreviada de describir cada instrucción y sólo son pura conveniencia para el entendimiento humano. El Spectrum no podría reconocer los nemónicos, excepto por mediación de un programa ensamblador.

El listado de ciertos convenios es el que sigue a continuación:

1) Nos referimos a los registros sencillos por una sola letra; por ejemplo, b. Los registros duales se expresarán en orden alfabético; por ejemplo, bc.
2) Un desplazamiento, d, se toma como positivo si está en el rango de 0 a 127 y negativo si pertenece al rango 128 al 255. No están permitidos los números mayores o menores que los citados.

   Un valor negativo se calcula mediante la operación de restar a 256 el desplazamiento, d. Por ejemplo, tomemos el caso de una instrucción del tipo "salto relatico sin condiciones" (Unconditional Relative Jump):

   jr d

   esta instrucción causa un salto hacia adelante en el programa de 8 octetos si el valor de *d* es 8 y un salto hacia atrás en el programa de ocho octetos si el valor para *d* es de 248 (256—8 = 248).

   Recuerde que cuando calcule un desplazamiento, el salto se hace desde la dirección del primer octeto siguiente a la instrucción.
3) Un número sencillo de un octeto, n, comprendido entre 0 y 255 inclusive.
4) Un número o dirección de dos octetos se representa por nn y estará en el rango de 0 a 65535 inclusive. El valor se calcula sumando el primer octeto, n, con 256 veces el segundo octeto.
5) nn entre paréntesis, por ejemplo (nn), significa "el contenido de la posición de memoria cuya dirección viene expresada por nn"; sin embargo, nn sin paréntesis significa "el número nn".

   Luego:

   ld hl, (23627)

   significará: "carga" el registro hl con el contenido de las direcciones 23627 y 23628, mientras que decir:

   ld hl, 23627

   significará: "carga" el registro hl con el número 23627. Análogamente (hl) significa "el contenido de la posición indicada por la dirección contenida en hl", mientras que hl sin paréntesis significa "el número en hl".

6) Siempre se expresará primero el destinatario del resultado de una instrucción; por ejemplo:

add a,b

que significa "Suma el contenido de *b* al contenido de *a* y deja el resultado en a".

Glosario de Instrucciones en Código Máquina

Esta sección presenta un sumario con la mayoría del "juego de instrucciones característico" del Z80A. Se han omitido algunas de las instrucciones más especializadas, como son las que se refieren al tratamiento con "interrupciones" (Interrupts), etc.

Tipo de instrucción — *Nemónico*

"No Operación" (No Operation) — nop

Esta es la instrucción maś simple y como su nombre indica, el procesador no hace nada cuando la encuentra. A menudo utilizaremos este tipo de instrucción cuando estemos "depurando" (Debugging) una rutina. Debido a que se puede sustituir temporalmente por una instrucción que nos resulte sospechosa de no trabajar correctamente, sin alterar el funcionamiento del resto de la rutina. También se puede usar este tipo de instrucción para introducir espacios en blanco entre instrucciones, por ejemplo, cuando se hacen pequeñas alteraciones a los programas existentes o para producir un retardo en una ejecución en particular si está incluída en un "lazo" (loop). El código, en decimal, es el 0 (cero).

"Carga" (Load) — ld

Las instrucciones de "carga" se usan para mover uno o dos octetos entre registros y entre registros y memoria.

Existen más de cien tipos distintos de instrucciones de "carga", más que cualquier otro tipo de instrucciones pertenecientes a la clase simple. Las instrucciones de carga estarán incluídas entre los ocho grupos siguientes:

1) 8 bit registro a registro.
   El contenido de cualquiera de los registros a, b, c, d, e, h ó l puede ser transferido de uno a otro.

2) 8 bit memoria a registro.
   (hl), (ix + d) ó (iy + d) pueden transferirse a cualquiera de los registros a, b, c, d, e, h ó l. Tanto (bc), como (de) o (nn) pueden transferirse al registro a.

3) 8 bit registro a memoria.
   a, b, c, d, e, h ó l pueden transferirse a (hl), (ix + d) ó (iy + d). El registro *a* se puede transferir a (bc), (de) ó (nn).

4) 8 Bit registro a memoria, en modo inmediato.

Se llama "inmediato" a un número leído desde el programa mismo en vez de leerse desde un registro o desde otra dirección en memoria. Un número, n, puede cargarse en a, b, c, d, e, h, l, (hl), (ix+d) ó (iy+d).

5) 16 bit registro a registro.
El contenido de hl, ix ó iy puede ser copiado en sp.

6) 16 bits memoria a registro.
(nn) puede transferirse a bc, de, hl, ix, iy, sp.

7) 16 bits registro a memoria.
bc, de, hl, ix, iy ó sp pueden transferirse a (nn).

8) 16 bits registro modo inmediato.
nn puede "cargarse" en bc, de, hl, ix, iy ó sp.

Instrucciones de "Empujar" (Push) y "Hacer Saltar" (Pop) push, pop

Una instrucción de "empujar" transfiere el contenido de un registro dado de 16 bits al "área de conservación de registros" y decrementa el "puntero del apilamiento" por dos veces. Una instrucción de "sacar" o "hacer saltar" hace justamente a la inversa, de manera que, se pueden usar dos instrucciones para guardar los valores de los registros y volver a cargarlos en el programa con posterioridad. Los registros duales af, bc, de, hl, ix e iy pueden ser cada uno de ellos "empujados y sacados".

"Intercambio" (Exchange) ex

Se pueden ejecutar instrucciones de "intercambio" entre hl y de, hl y (sp), ix y (sp), iy con (sp), af y af', y entre bcdehl y bcdehl' (una simple instrucción intercambia todos los registros de 8 bits).

"8 Bits Suma y Resta" add, sub, etc.

a, b, c, d, e, h, l, (hl), n, (ix+d) e (iy+d) pueden ser sumados o restados a/desde el registro "a", con o sin el "indicador de acarreo". Las instrucciones que involucran al "indicador de acarreo" terminan en c.

8 Bits "Y", "O" y "O-Exclusivo" (And, Or, Xor) and, or, etc

Los registros a, b, c, d, e, h, l, (hl), n, (ix+d) e (iy+d) pueden combinarse con el registro a, usando cualquiera de los tres operadores lógicos. El operador lógico "y" (And) activa cada bit en el resultado que figura activo en ambas "fuentes". El operador "O" (Or) activará todo bit en el resultado que lo estuviese en una o en ambas fuentes. El operador lógico "O-Exclusivo" (XOR) activará todo bit en el resultado que estuviese activo en una u otra fuente pero no activará los que estaban activos en ambas a la vez.

"Comparar" cp

Comparar es como restar, con la excepción de que sólo los indicadores y no el contenido del registro "a" van a verse afectados. Registros tales como a, b, c, d, e, h, l, (hl), n, (ix+d) e (iy+d) pueden ser comparados con el acumulador.

8 Bits, "Incrementar" y "Decrementar" inc, dec

Los registros a, b, c, d, e, h, l, (hl), (ix+d) e (iy+d) pueden ser incrementados o decrementados.

16 Bits "Suma" y "Resta" add, sub, etc.

bc, de, hl, ix pueden sumarse con o sin acarreo, también pueden restarse pero con acarreo solamente, estas dos operaciones se pueden hacer a/desde hl. Los registros bc, de, sp, ix, se pueden sumar sin acarreo al registro ix. Los registros bc, de, sp, iy pueden sumarse sin acarreo al registro iy.

"Saltar", "Llamar" y "Regresar" (Jump, Call, Return)

El registro indicador, f, contiene un indicador de acarreo, c; un indicador de paridad, p, que estará activo si el resultado es de paridad par; un indicador de signo, s, que estará activo si el valor es negativo; y un indicador de rebosamiento, v, que estará activo cuando haya rebose; y un indicador cero, z, que estará activo si el resultado fue un cero o una comparación con resultado de igualdad. Estos indicadores pueden usarse para controlar "saltos", "llamadas" (a subrutinas) y "regreso" (de subrutina).

Vamos a tratarlos por separado:

1) "Saltar" (Jump) jp ó jr

Los "saltos" a una dirección dada por nn, son posibles en los siguientes casos: "Salto absoluto" (jp); "Saltar si hay un cero o si no hay cero" (jpz) y (jpnz); "Saltar si hay acarreo o si no hay acarreo" (jpc) y (jpnc); "Saltar si es negativo o positivo" (jpp) y (jpm); "Saltar si p/v=1 o p/v=0" (jppe) y (jppo).

Los siguientes son "saltos relativos" hasta una dirección relativa (dada por d) desde la posición actual. Donde "d" estará comprendida en el rango de —128 a 127. Estos "saltos relativos" estarán comprendidos en estos casos:

a) "Salto absoluto relativo" (jr)
b) "Salto relativo si existe un cero o si no existe" (jrz ó jrnz).
c) "Salto relativo si hay acarreo o no lo hay" (jrc y jrnc).

También se pueden ejecutar "saltos" a las direcciones contenidas en hl,ix, ó iy (hl), jp (ix), jp (iy)). La instrucción dnjz decrementa el registro b y "salta" a d si b no es cero.

2) "Llamar" (Call) call

Esta instrucción ejecuta una acción similar a la ejercida por un comando descrito en Basic, cual es el llamado "GOSUB".

Si se establece la condición de "llamar" el programa transfiere control a la instrucción contenida en la dirección nn.

Están permitidas las siguientes "llamadas":
"Llamada absoluta" (Call); "Llamar si existe o no un cero" (Call z y call nz); "Llamar si hay o no acarreo" (call c y call nc); "Llamar si es positivo o negativo" (call p y call m); "Llamar si p/v = 1 o p/v = 0" (call pc y call po).

3) "Regresar" (Return) ret

Este tipo de instrucción tiene una función similar al comando en Basic llamado "RETURN". Las condiciones de "regreso" se establecen de manera que cada "llamada" ha de ir acompañada de un "regreso". También puede ejecutar (la inst. "regreso") desde una "interrupción" y desde una "interrupción no enmascarable" (Non-maskable interrupt).

Instrucciones de manejo de bits

Hemos visto que los ocho bits de cada registro se numeran del 0 al 7 de derecha a la izquierda. Cada una de las siguientes operaciones se puede ejecutar sobre los registros a, b, c, d, e, h, l y sobre (hl), (ix + d) e (iy + d).

Hay varias clases de manejo de bits:

1) "Prueba del Bit" (Bit Test) bit

Las instrucciones de "Prueba del Bit" activan el indicador de cero de forma opuesta a cómo esté emplazado el bit que se quiere probar (estado activo o no). Puede "probarse" cualquier bit.

2) "Activamiento de un bit" (Bit Set) set

Se puede activar cualquier bit.

3) "Desactivamiento de un bit" (Bit Reset) reset

Se puede ya desactivar cualquier bit.

4) "Rotar a la izquierda" rl

El bit 7 se transfiere al lugar que ocupa el bit de acarreo, éste a su vez se transfiere al bit 0 y el resto se transfieren un lugar a la izquierda.

5) "Rotar a la derecha" rr

El bit 0 se transfiere al lugar que ocupa el de acarreo, el de acarreo se transfiere el bit 7 y el resto se transfieren un lugar a la derecha.

6) "Rotar circularmente a la izquierda" rlc

El bit 7 se transfiere al mismo tiempo al lugar que ocupa el bit de acarreo y al bit 0. Todos los demás bits se transfieren un lugar a la izquierda.

7) "Rotar circularmente a la derecha" rrc

El bit 0 se transfiere al mismo tiempo al bit del acarreo y al bit 7. El resto de los bits se transfieren un lugar a la derecha.

8) "Desplazamiento aritmético a la izquierda" sla

Todos los bits se transfieren un lugar a la izquierda, el bit 7 se transfiere al lugar que ocupa el bit del acarreo y el bit 0 es desactivado.

9) "Desplazamiento aritmético a la derecha" sra

Todos los bits se transfieren un lugar a la derecha, el bit 0 se transfiere al del acarreo y el bit 7 se transfiere a sí mismo.

10) "Desplazamiento lógico a la derecha" srl

Es idéntico al desplazamiento aritmético a la derecha (ver apartado 9) excepto que el bit 7 está desactivado.

"Rotar un Dígito a la Izquierda" rld

Los bits 0 al 3 del registro a se transfieren a los bits 0 al 3 del registro dual (hl); bits 0 al 3 del (hl) se transfieren a los lugares de los bits 4 al 7 de (hl); bits 4 al 7 de (hl) se transfieren a los lugares de los bits 0 al 3 del registro a.

"Rotar un Dígito a la Derecha" rrd

Los bits 0 al 3 del registro a son transferidos a los bits 4 al 7 de (hl); bits 4 al 7 de (hl) se transfieren a los bits 0 al 3 de (hl); bits 0 al 3 de (hl) se transfieren a los bits 0 al 3 del registro a.

"Operaciones con el acumulador"

1) "Complementar" el registro a cpl

Todo bit que se encuentre en estado activo en a será desactivado y todo el que esté desactivado, pasará al estado activo.

2) "Negar" a neg

Esta operación consiste en:

a) complementar a.
b) sumarle un 1.

3) "Complementar el bit de acarreo" cpl

Activa el "indicador de acarreo" si éste está desactivado y a la inversa.

4) "Activación del bit de acarreo" sct

Activa sin más el indicador de acarreo.

5) "Ajuste decimal"

Corrige el registro a después de una suma o resta con bcd.

"Instrucción de volver a comenzar" (Restart)

Guarda el contador de programa en el "área de conservación de registros" y salta a la posición dada por 8*n, dónde n vendrá dado por el octeto inmediato siguiente.

"Manejo de bloques"

El cometido de estas instrucciones de tipo compuesto es el de mover datos o buscarlos en memoria. La clasificación es la siguiente:

1) "Cargar e incrementar" ldi

Mueve un octeto desde (hl) a (de). Incrementa hl y de y decrementa bc.

2) "Cargar, incrementar y repetir" ldir

Mueve un octeto desde (hl) a (de). Incrementa hl y de y decrementa bc. Repite esta operación hasta que bc = 0.

3) "Cargar y decrementar" ldd

Mueve un octeto desde (hl) a (de) y decrementa hl, de y bc.

4) "Carga, decrementa y repite" lddr

Mueve un octeto desde (hl) a (de) y decrementa hl, de y bc. Repite la operación hasta que bc = 0.

5) "Comparar e incrementar" cpi

Compara a y (hl). Incrementa hl y decrementa bc.

6) "Compara, incrementa y repite" cpir

Compara el registro a con (hl). Incrementa hl y decrementa bc. Repite hasta que a = (hl) ó bc = 0.

7) "Comparar y decrementar" cpd

Compara el registro a con (hl). Decrementa hl y bc.

8) "Compara, decrementa y repite" cpdr

Compara el registro a y (hl). Decrementa hl y bc. Repite hasta que a = (hl) ó bc = 0.

Sección B

4. INTRODUCCIÓN

Las 40 rutinas en código máquina que hay en esta sección están listadas en formato normalizado para permitir un manejo más fácil. Esta introducción explica el formato y presenta un programa en Basic, que puede usarse para cargar las rutinas en memoria.

Longitud:

Es la longitud de la rutina en octetos.

Número de Variables:

La ejecución de alguna de estas rutinas se puede controlar por medio de la alteración de los valores de una o más variables que pasaron a la rutina a través de la "memoria interna de la impresora" (Printer Buffer).

"Prueba de la suma" (Check sum)

Cada rutina se presenta como una secuencia de números enteros positivos que van a ser "introducidos" (mediante "POKE") en sucesivas posiciones de memoria. La prueba de la suma, (que consiste en sumar todos los números que integran la rutina) se da para que el usuario se asegure de que tiene cargada la rutina correctamente.

Operación:

Se da una breve explicación de la tarea ejecutada por la rutina.

Variables:

Se definen los nombres, longitud y posición de cada variable dentro de la "memoria intermedia de la impresora". Una variable con un octeto de longitud debe ser un número entero y positivo tomado entre 0 y 225 inclusive, y se enviará desde el Basic o desde el teclado por medio de:

POKE *posición, valor*

Una variable de dos octetos pasará usando dos comandos:

POKE *posición, valor*—256*INT(*valor/256)*
POKE *posición* + 1, INT (*valor*/256)

Las posiciones usadas están en la memoria intermedia de impresora (Printer buffer).

Llamada (Call):

Se llama a las rutinas por medio de la función USR que ha de incorporarse al comando. Si la rutina en código máquina no reenvía el valor al Basic, se recomienda el uso del comando "RAND" como se muestra a continuación:

RAND USR dirección

Si el valor en el registro bc va a ser reenviado, entonces usar:

LET A = USR dirección

o tambien:

PRINT USR dirección

que es más recomendable, dependiendo de en dónde se va a almacenar el valor devuelto (o reenviado), si en una variable en Basic o mostrado en pantalla.

"Comprobación de errores" (Error Checks):

Serán explicadas también las comprobaciones hechas por la rutina sobre operaciones ilógicas (sin sentido) o valores de variables en conflicto, etc.

Comentarios:

Se explicarán ciertas variantes simples a la rutina principal.

Listado en Código Máquina:

Se presenta la rutina en lenguaje de ensamblamiento con forma absoluta en la tercera columna, encabezada por "números a introducir". Para cargar la rutina se deben introducir en memoria (POKE) los números de la tercera columna, guardando su secuencia. Todos los números están en decimal.

Funcionamiento:

Se explica el modo de operación de la rutina haciendo referencia al listado en código máquina.

Cargador para Código Máquina (Cargador M/C)
(M/C Loader)

Casi todas las rutinas en código máquina de este libro son "relocalizables" es decir, deberan funcionar sin importar dónde están localizadas dentro de la memoria RAM. Si una rutina no es "relocalizable" encontraremos en el párrafo de comentarios la explicación de cómo deberemos modificarla para almacenarla en una posición distinta de la implementada en su diseño.

Hemos visto en la Sección "A", Capítulo 2, que el Spectrum usa varias partes de la RAM para funciones distintas y que el área existente entre la posición indicada por las "variables del sistema" "RAMTOP" y "UDG" es para el almacenamiento de las rutinas en código máquina.

El programa titulado "BP" se puede usar para cargar, alterar y mover una rutina en código máquina. Mediante éste, el usuario puede desactivar el puntero "RAMTOP", con el fin de darle más espacio a la rutina; puede introducir una rutina desde el teclado, ir paso a paso hacia arriba o hacia abajo dentro de la rutina para poder corregir un error, insertar o cancelar partes de ésta.

Cuando el programa se pone en marcha (RUN) éste imprime (PRINT) la dirección más baja en la que podemos almacenar la rutina, por ejemplo, una maś que la de "RAMTOP", y la cantidad de espacio disponible entre esa dirección y el final de la RAM.

En las máquinas de 16 K la dirección más baja es 32600, aunque el usuario tenga alterada la variable del sistema "RAMTOP". De manera similar, en la máquina de 48 K la dirección más baja normalmente es 65368.

Los 168 octetos del final de la RAM están reservados normalmente para gráficos definidos por el usuario. Pero el programa le permite disponer de este área si así lo quiere. Alternativamente puede disponer el usuario de una nueva dirección aún más baja, en la cual el programa pone luego el puntero del "RAMTOP" usando el comando "CLEAR". El programa no aceptará una dirección más baja de 27000 debido a que la rutina inundaría el espacio que se requiere para el programa propiamente dicho. El programa pregunta por la dirección en la cual arranca la rutina. Por tanto el usuario puede reservar espacio para ciertas rutinas que después las cargará cada una por separado.

Toda vez que el usuario ha tenido ya la oportunidad de cambiar su selección si no quedó satisfecho, el programa imprime la información gráfica principal. La figura BF1 nos muestra la forma que va a tener la información gráfica, cuando la rutina de "pantalla invertida" (Screen Invert) ha sido cargada en la dirección 32000. La primera columna corresponde a la dirección, la segunda es el contenido de la dirección y la tercera es la "prueba de la suma". La rutina llamada "pantalla invertida" tiene una longitud de 18 octetos y la prueba de la suma resulta ser 1613. Esta rutina ocupará las posiciones 32000 a la 32017 y la prueba de la suma para la posición 32017, o lo que es igual, la suma de los contenidos de las posiciones 32000 a la 32017, es 1613.

Cuando la información principal se muestra al usuario, su atención se concentra en una posición porque el contenido decimal relampaguea (Flash). A esto se le llama la posición actual (Current Location) e inicialmente ésta es la dirección elegida de comienzo de la rutina. El usuario introduce un número entero entre 0 y 255 inclusive, que el programa a su vez introduce (POKE) en la posición actual y luego la siguiente dirección viene a ser la posición actual. De esta manera una rutina completa puede ser introducida (POKE) en un lugar, la información gráfiuca se acualiza y si es necesario hace aparecer una nueva información en pantalla (SCROLL)(1) por cada paso.

El usuario puede elegir no introducir un número, sino seleccionar una opción de entre las enumeradas en la tabla BT1. Estas propiedades permiten hacer correcciones.

Código	*Opción*
b	Mueve la posición actual una dirección hacia atrás.
b número	Mueve la posición actual un número de direcciones (dado por el número que especifiquemos) hacia atrás.
f	Mueve la posición actual una dirección hacia adelante.
f número	Mueve la posición actual un número de direcciones (dado por el número que especifiquemos) hacia adelante.

(1) N. del T.: Este término (SCROLL) de difícil traducción lo vamos a conservar como tal a lo largo del libro. Sin embargo es muy conveniente saber cómo trabaja y el efecto que produce. Fundamentalmente, es un giro de la visualización consistente en entrar por la parte superior o inferior de la pantalla nuevas líneas de visualización (de la información) en el área de visión. También por ambos lados de la pantalla (derecha e izquierda).

i número	Inserta un cierto número de octetos, (especificado por el número que pongamos) conteniendo un cero cada uno de ellos, en la posición actual.
d número	Cancela un cierto número de octetos (dado por número) en la posición actual.
t	Pone fin al programa.

Tabla BT1. *Opciones disponibles para editar en código máquina*

```
100 GO SUB 8100
200 REM*****Calcula la memoria disponible
210 LET min = 1 + PEEK 23730 + 256*PEEK 23731
220 LET p = PEEK 23732 + 256*PEEK 23733
230 LET t = p—min + 1
400 REM*****Tomar dirección de comienzo
410 PRINT "Dirección más baja posible de comienzo = ";min,,,
"Máximo espacio disponible = ";t
420 INPUT "Desea cambiar la dirección más baja?
(S o N)"; Z$
430 IF z$ = "S" OR z$ = "s" THEN GO TO 7000
440 INPUT "Introducir la dirección de comienzo de la carga del
código máquina.... ";a
450 IF a < min OR a > p THEN BEEP . 2,24: GO TO 440
500 GO SUB 8100
510 LET t = t—a + min
520 PRINT "Puede usar hasta" ; t ; "octetos" , , ,
530 LET u = PEEK 23675 + 256*PEEK 23676
540 IF a<u AND u<p THEN PRINT "Si usa más de" ; u—a ;
"octetos, debe anular el área de gráficos de usuario"
550 IF a > u THEN PRINT "Escribirá sobre el área de gráficas
definidos por el usuario."
560 INPUT "Es eso correcto (S ó N)?"; z$
570 IF z$ = "N" OR z$ = "n" THEN GO TO 7000
580 IF z$ < >"S" AND z$< > "s" THEN BEEP. 2,24: GO TO 560
700 REM ***** adelante y carga
710 LET i = a
750 GO SUB 8200
760 INPUT "Introducir número ,b,f,i,d o t " ; z$
770 IF z$ = " " THEN BEEP . 2,24: GO TO 760
780 LET a$ = CHR$ (CODE z$(1)—32*(z$(1) > "£ "))
790 GO TO 800 + 200*(a$ = "B") + 300*(a$ = "f") + 400*(a$ = "I")
+ 500*(a$ = "D") + 600*(a$ = "T")
800 LET x = VAL z$
810 IF i>p THEN BEEP . 2,24: GO TO 750
820 IF x<0 OR x>255 OR x<>INT x THEN BEEP .2,24: GO TO
760
830 POKE i,x
```

```
840 LET i=i+1
850 GO TO 740
1000 REM ***** mueve hacia adelante
1010 LET i=i—1
1020 IF LEN z$>1 THEN LET i=i+1—VAL z$ (2 TO )
1030 IF i<a THEN LET i=a
1040 GO TO 740
1100 REM ***** mueve hacia atrás
1110 LET i=i+1
1120 IF LEN z$>1 THEN LET i=i—1+VAL z$ (2 TO )
1130 IF i>p THEN LET i=p
1140 GO TO 740
1200 REM ***** inserta
1210 IF LEN z$=1 THEN LET n=1: GO TO 1225
1220 LET n=VAL z$(2 TO ): IF n<1 OR n>p—i OR n<>INT n
THEN BEEP . 2,24: GO TO 740
1225 CLS : GO SUB 8100: PRINT TAB 6; "Se está insertando"
1230 FOR j=p TO i+n STEP —1
1240 POKE j,PEEK (j-n)
1250 NEXT j
1260 FOR j=i TO i+n—1
1270 POKE j,0
1280 NEXT j
1290 GO TO 740
1300 REM ***** Borra
1310 IF LEN z$=1 THEN LET n=1: GO TO 1330
1320 LET n=VAL z$ (2 TO ) : IF n<1 OR n>p—i OR n<>INT n
THEN BEEP . 2,24: GO TO 740
1330 IF n<0 OR n>p—i THEN BEEP . 2,24: GO TO 1320
1340 CLS : GO SUB 8100: PRINT TAB 6; "Se está eliminado"
1350 FOR j=i TO p—n
1360 POKE j,PEEK (j+n)
1370 NEXT j
1380 GO TO 740
1400 STOP
1401 PRINT AT 21,7: "Programa terminado"
1410 STOP
7000 REM ***** desactiva el RAMTOP
7010 INPUT "Introducir nueva dirección de comienzo" ; a
7020 IF a<27000 OR a>p THEN BEEP . 2,24: GO TO 7010
7030 CLEAR a—1
7040 RUN
7999 STOP
8100 CLS
8110 PRINT TAB 4; "Cargador del código máquina" , , ,
8120 RETURN
8200 REM ***** muestra memoria
8210 GO SUB 8100
```

```
8220 PRINT "Dirección   decimal   prueba   de la suma"
8230 LET c=0
8240 LET s=i—8: IF s< a THEN LET s=a: GO TO 8280
8250 FOR j=a TO s—1
8260 LET c=c+PEEK j
8270 NEXT j
8280 LET f=s+17: IF f>p THEN LET f=p
8290 FOR j=s TO f
8300 LET c=c+PEEK j
8310 PRINT AT j—s + 3,5; j; TAB 12; PEEK j; TAB 22;c
8320 NEXT j
8400 LET pos=i—s+3
8410 PRINT AT pos, 12; FLASH 1; PEEK i
8420 RETURN
```

Cargador del Código Máquina

Direccion	*Decimal*	*Prueba de la suma*
32000	33	33
32001	0	33
32002	64	97
32003	1	98
32004	0	98
32005	24	122
32006	22	144
32007	255	399
32008	122	521
32009	150	671
32010	119	790
32011	35	825
32012	11	836
32013	120	956
32014	177	1133
32015	32	1165
32016	247	1412
32017	201	1613

Figura BF.1 *Información gráfica producida por el cargador del Código Máquina cuando se ha cargado en la posición 32000 la rutina de "inversión de pantalla"*

5. RUTINAS DE "SCROLL"(1)

"SCROLL" de las Cualidades por la Izquierda

Longitud: 23
Número de variables: 1
Prueba de la suma: 1574

Operation

Esta rutina hace "scroll" de las cualidades de todos los caracteres en pantalla una posición.

Variables

Nombre	*Longitud*	*Dirección*	
nueva cualidad	1	23296	La cualidad a meter en la columna más a la derecha

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error checks):

Ninguna

Comentarios:

Esta rutina se usa frecuentemente para hacer destacar zonas de texto y gráficos. Para "representar en pantalla" (scroll) solamente las 22 líneas superiores, (de las 24 que tiene la pantalla) deberá cambiarse el número 24 que lleva un asterisco (24*), en el listado de código máquina (columna de "números a introducir"), por el número 22.

Listado de Código Máquina

Etiqueta	*Lenguaje de Ensamblador*	*Números a introducir*
	ld hl, 22528	33 0 88
	ld a, (23296)	58 0 91
	ld c, 24	14 24*
siguiente línea	ld b, 31	6 31
siguiente carácter	inc hl	35
	ld e, (hl)	94
	dec hl	43
	ld (hl),e	115
	inc hl	35

djnz, next char	16 249
ld (hl),a	119
inc hl	35
dec c	13
jr nz, next line	32 242
ret	201

Funcionamiento:

El registro dual hl se carga con la dirección del "área de cualidades". El registro acumulador se carga con el valor de la cualidad, que será introducida en la columna situada en la derecha. El registro c estará cargado con el número de líneas que han de ir apareciendo en pantalla (Scroll), de modo que pueda usarse como un contandor de líneas. El registro b tomará un valor igual al número de caracteres por línea menos uno, a fin de usarse como un contador.

El registro hl es incrementado para apuntar a la siguiente cualidad y ésta se carga en el registro e. El registro hl se decrementa y se le introduce (POKE) el valor que hay en el registro e. El registro dual hl se incrementa de nuevo para apuntar a la siguiente cualidad. El registro se decrementa y si este registro no contiene un cero todavía se ejecutaa un salto hacia atrás en el programa hasta la etiqueta "siguiente carácter". El registro hal apunta ahora a la columna derecha y éste se carga con el valor contenido en el acumulador. De nuevo el registro hl se incrementa hasta apuntar al comienzo de la siguiente línea. El contador de líneas, que establecimos en el registro c, se decrementa. Si el valor resultante no es cero, la rutina ejecuta el lazo que comienza en la etiqueta "siguiente línea".

Una vez ejecutada la rutina, ésta devuelve el control al Basic.

"SCROLL" de las Cualidades por la Derecha

Longitud: 23
Número de variables: 1
Prueba de la suma 1847

OPERACION:

Esta rutina hace "scroll" (hace aparecer en pantalla . . .) de las cualidades de todos los caracteres, en pantalla hacia su parte derecha, carácter por carácter.

"Variables"

Nombre	*Longitud*	*Posición*	*Comentario*
nueva cualidad	1	23296	La cualidad a meter en la columna más a la izquierda

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Ninguna

Comentarios:

Esta rutina se usa para resaltar áreas de texto y gráficos. Para "representar en pantalla" (Scroll) solamente las 22 líneas superiores, deberá cambiarse el número 24 que lleva el asterisco (24*) en el listado por el número 22.

Listado de Código Máquina

Etiqueta	*Lenguaje de ensamblador*	*Números a introducir*
	ld hl, 23295	33 255 90
	ld a, (23296)	58 0 91
	ld c, 24	14 24*
siguente línea	ld b, 31	6 31
siguiente carácter	dec hl	43
	ld e, (hl)	94
	inc hl	35
	ld (hl),e	115
	dec hl	43
	djnz, next char	16 249
	ld (hl),a	119
	dec hl	43
	dec c	13
	jr nz, next line	32 242
	ret	201

Funcionamiento:

El registro dual hl se carga con la dirección del último octeto del área de cualidades. El acumulador se carga con el valor de la cualidad a introducir en la columna de la izquierda. El registro c se carga con el número de líneas que se "presentarán en pantalla" (scroll), de manera que este registro pueda usarse como un contador de líneas. El registro b tomará un valor igual al número de caracteres por línea menos uno, a fin de usarlo como contador.

El registro dual hl se decrementa para que apunte a la siguiente cualidad. El valor de dicha cualidad se carga en el registro e. El registro hl se incrementa y luego se cargará (POKE) con el valor del registro e. De nuevo se decrementa hl para apuntar a la cualidad siguiente. El contador que hay en el registro b se decrementa, y si este todavía no contiene un cero, la rutina retrocede hasta la etiqueta "siguiente carácter".

El registro hl apunta ahora a la columna más a la izquierda, y éste se carga con el valor que hay en el acumulador. El contador de líneas se decrementa, y si éste no contiene un cero la rutina retrocede al principio del lazo cuya eti queta indica "siguiente línea".

La rutina devuelve el control al Basic una vez ejecutada.

"SCROLL" de las Cualidades en Sentido Ascendente

Longitud: 21
Número de variables: 1
Prueba de la suma: 1591

Operación

Esta rutina "introduce en pantalla" (Scroll) las cualidades de todos los caracteres en sentido ascendente, carácter a carácter.

"Variables"

Nombre	*Longitud*	*Dirección*	*Comentario*
Nueva cualidad	1	23296	La nueva cualidad a meter en la línea inferior

Llamada

RAND USR dirección

"Comprobación de errores" (Error Checks)

Esta rutina hace destacar áreas de texto gráficos. Para "introducir en pantalla" las 22 líneas superiores, cambiar el valor afectado con un asterisco (columna de "números a introducir", valor 224*) por el nuevo valor 160.

Listado de Código Máquina

Etiqueta	*Lenguaje Ensamblador*	*Números a introducir*
	ld hl, 22560	33 32 88
	ld de, 22528	17 0 88
	ld bc, 736	1 224* 2
	ldir	237 176
	ld a, (23296)	58 0 91
	ld b, 32	6 32
Siguiente carácter	ld (de),a	18
	inc de	19
	djnz next char	16 252
	ret	201

Funcionamiento:

El registro hl se carga con la dirección de la segunda línea de cualidades, el registro *de* se carga con la dirección de la primera línea y el registro *bc* se carga con el número de octetos que se van a mover.

Los *bc* octetos que comienzan en los señalados por *hl* se copian en los señalados por *de*, usando la instrucción "ldir". El resultado obtenido en el registro de apunta a la línea inferior de cualidades. El acumulador se carga con el código de la cualidad que va a entrar por la línea inferior de pantalla. El regis-

tro *b* se carga luego con el número de caracteres que haya en la línea, de manera que se pueda usar como contador.

El registro *de* se le introduce (Poke) el valor del acumulador, luego se incrementa para que apunte al octeto siguiente. El contador se decrementa, y si éste no contiene aún el valor cero, la rutina retrocede al principio del lazo cuya etiqueta es "siguiente carácter". La rutina devuelve el control al Basic una vez ejecutada.

"SCROLL" de las Cualidades en Sentido Descendente

Longitud: 21
Número de variables: 1
Prueba de la suma: 2057

Operación

Esta rutina "introduce en pantalla" (Scroll) las cualidades de todos los caracteres en sentido descendente, carácter a carácter.

"Variables"

Nombre	*Longitud*	*Posición*	*Comentarios*
Nueva cualidad	1	23296	La cualidad a meter en la línea superior.

Llamada

RAND USR dirección

"Comprobación de errores" (Error Checks)

Ninguna

Comentarios:

Esta rutina se usa para destacar áreas de texto y gráficos. Para "introducir en pantalla" (Scroll) las 22 líneas superiores, se han de introducir los siguientes cambios en los números que llevan asteriscos de la columna "números a introducir":

El número 223* cambiarlo por el 159
El número 255** cambiarlo por el 191
El número 224*** cambiarlo por el 160

Listado del Código Máquina

Etiqueta	*Lenguaje de ensamblador*	*Números a introducir*
	ld hl, 23263	33 223* 90
	ld de, 23295	17 255** 90
	ld bc, 736	1 224*** 2
	lddr	237 184
	ld a, (23296)	58 0 91

	ld b, 32	6 32
Siguiente carácter	ld (de),a	18
	dec de	27
	djnz next char	16 252
	ret	201

Funcionamiento:

El registro dual *hl* se carga con la dirección de la última cualidad que hay en la línea 23. El registro *de* se carga con la dirección de la última cualidad que hay en la línea 24. El registro *bc* se carga con el número de octetos que se van a mover. Luego, la instrucción "lddr" mueve los *bc* octetos terminando en los del registro *hl*, de modo que finalicen en el registro *de*. Esto da por resultado que *de* contenga la dirección de la última cualidad que hay en la primera línea.

El acumulador se carga con el valor de la cualidad que entra por la línea superior. El registro *b* se carga con el número de octetos que hay en la línea superior, con el fin de usarle como contador. Al registro *de* se le introduce el valor que haya en el acumulador, posteriormente el registro *de* se decrementa para que apunte al siguiente octeto. El contador se decrementa, y si no contiene todavía un cero, se ejecuta un salto al principio del lazo encabezado por la etiqueta "siguiente carácter".

La rutina vuelve luego al Basic.

"SCROLL" por la Izquierda Carácter por Carácter

Longitud: 21
Número de variables: 0
Prueba de la suma: 1745

Operación

Esta rutina "introduce en pantalla" (Scroll) el contenido de información gráfica" un carácter por la izquierda.

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Ninguna

Comentarios:

Esta rutina es útil cuando usamos la pantalla como una "mirilla " o "ventana" a través de la cual se muestra una pequeña región de una gran área de información gráfica. La "mirilla" la desplazaremos usando las rutinas de "scroll".

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, 16384	33 0 64
	ld a,l	85
	ld a,192	62 192
siguiente línea	ld b,31	6 31
siguiente octeto	inc hl	35
	ld e, (hl)	94
	dec hl	43
	ld (hl),e	115
	inc hl	35
	djnz next byte	16 249
	ld (hl),d	114
	inc hl	35
	dec a	61
	jr nz, next line	32 242
	ret	201

Funcionamiento:

El registro dual *hl* se carga con la dirección del "archivo de información gráfica", y el registro *d* se emplaza a cero. El acumulador se carga con el número de líneas en pantalla. El registro *b* toma el valor del número de caracteres por línea menos uno, o lo que es igual, el número de octetos que se van a copiar.

El registro *hl* se incrementa para que apunte al siguiente octeto, el registro *e* se cargará con este valor. El registro *hl* sedecrementa y toma el valor que hay en el registro *e*. El registro dual *hl* se incrementa para direccionar al siguiente octeto, el contador que hay en el registro *b* se decrementaa. Si éste no contiene el valor cero la rutina retrocede al principio del lazo cuya etiqueta es "siguiente octeto".

Si el registro *b* contiene un cero, el último octeto de la línea ha sido copiado, y el registro *hl* apunta al octeto más a la derecha. Luego, éste se carga con cero y el registro *hl* se incrementa para apuntar a la siguiente línea. El contador de línea que hay en el acumulador se decrementa y si no contiene un cero se ejecuta un salto a "siguiente línea".

La rutina vuelve luego al Basic.

"SCROLL" por la Derecha Carácter por Carácter

Longitud: 22
Número de variables: 0
Prueba de la suma: 1976

Operación

Esta rutina "introduce en pantalla" (Scroll) el contenido del "archivo de información gráfica" un carácter a la derecha.

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Ninguna

Comentario

Esta rutina es útil cuando se usa la pantalla como una "mirilla", de manera que muestre una pequeña región de un área grande de información gráfica. La "mirilla" se moverá usando rutinas de "scroll".

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, 22527	33 255 87
	ld d, O	22 O
	ld a, 192	62 192
siguiente línea	ld b, 31	6 31
siguiente octeto	dec hl	43
	ld e, (hl)	94
	inc hl	35
	ld (hl),e	115
	dec hl	43
	djnz next byte	16 249
	ld (hl),d	114
	dec hl	43
	dec a	61
	jr nz, next line	32 242
	ret	201

Funcionamiento:

El registro dual *hl* se carga con la dirección del último octeto del "archivo de información gráfica" y el registro *d* se emplaza a cero. El acumulador se carga con el número de líneas en pantalla. El registro *b* se carga con el número de caracteres por línea menos uno, para ser usado como contador.

El registro dual *hl* se decrementa para apuntar al siguiente octeto, y la celdilla que señale se carga en el registro *e*. El registro *hl* se incrementa luego y se le carga con valor del registro *e*. *hl* se decrementa para apuntar al siguiente octeto, y el contador del registro *b* se decrementa. Si el registro *b* no contiene un cero, la rutina retrocede al principio del lazo cuya etiqueta es "siguiente octeto".

Si el registro *b* contiene un cero, *hl* apunta al octeto más a la izquierda de la línea. Este se carga con cero, y *hl* se decrementa para apuntar a la siguiente

línea. El contador en el acumulador se decrementa luego y si no contiene un cero, se ejecuta un salta a "siguiente línea".

La rutina vuelve luego al Basic.

"SCROLL" en Sentido Ascendente Cáracter por Carácter

Número de variables: 0
Prueba de la suma: 6328

Operación

Esta rutina "introduce en pantalla" (Scroll) el contenido del "archivo de información gráfica" en sentido ascendente por ocho "elementos de imagen" (Pixels) (de ocho en ocho).

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Ninguna

Comentarios:

Ninguno

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, 16384	33 0 64
	ld de, 16416	17 32 64
"Save" (Guardar)	push hl	229
	push de	213
	ld c,23	14 23
"Next Line" (Siguiente línea)	ld b,32	6 32
"Copy Byte" (Copiar octeto)	ld a, (de)	26
	ld (hl),a	119
	ld a,c	121
	and 7	230 7
	cp 1	254 1
	jr nz, next byte	32 2
	sub a	151
	ld (de),a	18
"Next Byte" (Copiar octeto)	inc hl	35
	inc de	19

	djnz copy byte	16 241
	dec c	13
	jr z, restore	40 19
	ld a,c	121
	and 7	230 7
	cp O	254 O
	jr z, next block	40 22
	cp 7	254 7
	jr nz, next line	32 225
	push de	213
	ld de, 1792	17 0 7
	add hl,de	25
	pop de	209
	jr next line	24 217
"Restore" (Volver a almacenar)	pop de	209
	pop hl	225
	inc d	20
	inc h	36
	ld a,h	124
	cp 72	254 72
	jr nz, save	32 204
	ret	201
"Next Block" (Siguiente bloque)	push hl	229
	ld hl, 1792	33 0 7
	add hl,de	25
	ex de,hl	235
	pop hl	225
	jr next line	24 198

Funcionamiento:

El registro dual *hl* se carga con la dirección del "archivo de información gráfica", y el registro *de* se carga con la dirección del octeto situado ocho líneas abajo. El registro *hl* y el *de* se guardan en el "área de conservación de registros". El registro *c* se carga con el número de "líneas a imprimir" (Print lines) menos una, de la pantalla. El registro *b* se carga con el número de octetos en una línea de información gráfica, para ser usado como contador.

El acumulador se carga con el octeto direccionado por el registro *de* y éste es introducido (Poke) en el direccionado por *hl*. El acumulador se carga con el contenido del registro *c* y si éste contiene 1,9, ó 17 entonces el direccionado por *de* se carga con cero. Los registros *hl* y *de* se incrementan para que apunten a los octetos siguientes, el contador en el registro *b* se decrementa, y si éste no contiene un cero, se ejecuta un salto a "Copy Byte".

El contador de líneas *c* se decrementa luego. Si éste contiene un cero se ejecuta un salto a "Restore". Si el registro *c* no contiene 8 ó 16 se ejecuta luego un salto a "Next Block". Si *c* no contiene un 7 ni un 15, la rutina vuelve a ejecutar el lazo encabezado por "Next Line". el número 1792 se suma al registro

hl de modo que *hl* apunte al siguiente bloque de la pantalla. La rutinasalta luego a "Next Line".

En "Restore" los registros *de* y *hl* son recogidos del área de conservación de registros, sumándoles a cada uno 256. Así, los registros *de* y *hl* apuntan a una línea por debajo de la posición que ocupaban en el lazo anterior. Si el *hl* contiene 18432, la rutina vuelve al Basic, de lo contrario se ejecuta un salto a "Save". En "Next Block", se suma 1792 al registro *de* de manera que *de* apunte al siguiente bloque de la pantalla. La rutina luego vuelve a ejecutar el lazo encabezado por "Next Line".

"SCROLL" en Sentido Descendente Carácter por Carácter

Longitud: 73

Número de variables: 0

Prueba de la suma: 7987

Operación:

Esta rutina "introduce en pantalla" (Scroll) el contenido del "archivo de información gráfica" en sentido descendente por ocho "elementos de imagen".

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Ninguna

Comentarios

Ninguno

Listado del Código Máquina

Etiqueta	*Lenguaje Ensamblador*	*Números a introducir*
	ld hl, 22527	33 255 87
	ld de, 22495	17 223 87
"Save" (Guardar)	push hl	229
	push de	213
	ld c, 23	14 23
"Next Line" (siguiente línea)	ld b,32	6 32
"Copy Byte" (Copiar octeto)	ld a, (de)	26
	ld (hl),a	119
	ld a,c	121
	and 7	230 7
	cp 1	254 1
	jr nz, next byte	32 2
	sub a	151
	ld (de),a	18

"Next Byte" (siguiente octeto)	dec hl	43
	dec de	27
	djnz copy byte	16 241
	dec c	13
	jr z, restore	40 21
	ld a,c	121
	and 7	230 7
	cp O	254 O
	jr z, next block	40 24
	cp 7	254 7
	jr nz, next line	32 225
	push de	213
	ld de, 1792	17 0 7
	and a	167
	sbc hl,de	237 82
	pop de	209
	jr next line	24 215
"Restore" (volver a almacenar)	pop de	209
	pop hl	225
	dec d	21
	dec h	37
	ld a,h	124
	cp 79	254 79
	ret z	200
	jr save	24 201
"Next Block" (Siguiente bloque)	push hl	229
	ld hl, 1792	33 0 7
	ex de, hl	235
	and a	167
	sbc hl,de	237 82
	ex de, hl	235
	pop hl	225
	jr next line	24 193

Funcionamiento:

El registro dual *hl* se carga con la dirección del último octeto del "archivo de información gráfica", y el registro *de* se carga con la dirección del octeto situado ocho líneas por arriba. El *hl* y el *de* se guardan en el "área de conservación de registros". El registro *c* se carga luego con el número de "líneas a imprimir" (Print Lines) menos una, de la pantalla. El registro *b* se carga con el número de octetos que hay en una línea de información gráfica, para usarse como contador.

El acumulador se carga con el octeto direccionado por *de,* y éste se introduce en el direccionado por *hl.* El acumulador se carga con el contenido del registro *c,* y si éste contiene 1, 9, ó 17 se carga luego el direccionado por *de* con cero. Los Registros *hl* y *de* don entonces decrementados para apuntar a los si-

guientes octetos de la información gráfica. El contador en el registro *b* se decrementa y si éste no contiene un cero, se ejecutan un salto a "Copy Byte".

Se decrementa el contador de líneas en el registro *c,* y si éste contiene un cero se ejecuta un salto a "Restore". Si el registro *c* contiene 8 ó 16 se ejecuta un salto a "Next Block". Si el registro *c* no contiene 7 ó 15 la rutina vuelve al principio del lazo "Next Line". 1972 se resta del registro *hl,* de manera que *hl* apunta al siguiente bloque de la pantalla. La rutina salta a "Next Line".

"SCROLL" a la Izquierda por un Elemento de Imagen

Longitud: 17
Número de variables: 0
Prueba de la suma: 1828

Operación:

Esta rutina "introduce en pantalla" (Scroll) el contenido del "archivo de información gráfica" un elemento de imagen a la izquierda.

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Ninguna

Comentario:

Esta rutina da un movimiento más suave que la rutina "scroll a la izquierda carácter a carácter", pero se requiere ocho "llamadas" (Call) para mover la información gráfica un carácter completo.

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, 22527	33 255 87
	ld c, 192	14 192
"Next Line" (siguiente línea)	ld b, 32	6 32
	or a	183
"Next Byte" (siguiente octeto)	rl (hl)	203 22
	dec hl	43
	djnz next byte	16 251
	dec c	13
	jr nz, next line	32 245
	ret	201

Funcionamiento:

El registro dual *hl* se carga con la dirección del último octeto del "archivo de información gráfica", y el registro *c* se carga con el número de líneas que hay en el "archivo de información gráfica", para usarse como un contador de líneas. El registro *b* se carga con el número de octetos en una línea, para su uso como contador. El "indicador de acarreo" se emplaza a cero.

El octeto direccionado por *hl* se rota luego en un bit a la izquierda, el indicador de acarreo se copia en el bit más a la derecha, y el bit más a la izquierda se transfiere al que ocupa el bit del indicador de acarreo. El registro dual *hl* se decrementa para apuntar al octeto siguiente y el contador en el registro *b* se decrementa. Si éste no contiene un cero, la rutina vuelve a ejecutar el lazo "Next Byte". El número de líneas se decrementa, y si éste no es igual a cero la rutina salta hacia atrás a "Next Line".

La rutina vuelve luego al Basic.

"SCROLL" a la Derecha por un Elemento de Imagen

Longitud: 17
Número de varibles: 0
Prueba de la suma: 1550

Operación:

Esta rutina hace "scroll" del contenido del "archivo de información gráfica" un elemento de imagen a la derecha.

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Ninguna

Comentarios:

Esta rutina da un movimiento más suave que la llamada "scroll por la derecha carácter por carácter" pero se necesitan ocho "llamadas" (Call) para mover la información gráfica un carácter completo.

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, 16384	33 0 64
	ld c, 192	14 192
"Next Line" (siguiente línea)	ld b, 32	6 32
	or n	183
"Next Byte" (siguiente octeto)	rr (hl)	203 30
	inc hl	35

	djnz next byte	16 251
	dec c	13
	jr nz, next line	32 245
	ret	201

Funcionamiento:

El registro dual *hl* se carga con la dirección del "archivo de información gráfica", y el registro *c* se carga con el número de líneas que hay en la información gráfica, para usarse como un contador de líneas. El registro *b* se carga con el número de octetos que hay en una línea, para usarse como un contador. El indicador de acarreo se emplaza luego a cero. El octeto direccionado por *hl* se rota después un bit a la derecha, el indicador de acarreo se copiará en el bit más a la izquierda, y el bit más a la derecha será copiado en el del indicador de acarreo. El registro dual *hl* se incrementa para apuntar al siguiente octeto y el contador en el registro *b* se incrementa luego. Si éste no contiene un cero, la rutina retrocede a "Next Byte". El contador de líneas se decrementa, y si éste no es igual a cero la rutina salta atrás hasta "Next Line".

La rutina retorna al Basic.

"SCROLL" en Sentido Ascendente por un Elemento de Imagen

Longitud: 91
Número de variables: 0
Prueba de la suma: 9228

Operación:

Esta rutina hace "scroll" del contenido del "archivo de información gráfica" en sentido ascendente, por un elemento de imagen (uno por uno).

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Ninguna

Comentarios:

Ninguno

Listado del Código Máquina

Etiqueta	*Lenguaje Ensamblador*	*Números a introducir*
	ld hl, 16384	33 0 64
	ld de, 16640	17 0 65
	ld c, 192	14 192
"Next Line" (siguiente línea)	ld b, 32	6 32

"Next Byte" (siguiente octeto)	ld a, (de)	26
	ld (hl),a	119
	ld a,c	121
	cp 2	254 2
	jr nz, next byte	32 2
	sub a	151
	ld (de),a	18
"Next Byte" (siguiente octeto)	inc de	19
	inc hl	35
	djnz copy byte	16 243
	push de	213
	ld de, 224	17 224 O
	add hl,de	25
	ex (sp),hl	227
	add hl,de	25
	ex de,hl	235
	pop hl	225
	dec c	13
	ld a,c	121
	and 7	230 7
	cp O	254 O
	jr nz, subtract	32 10
	push de	213
	ld de, 2016	17 224 7
	and a	167
	sbc hl,de	237 82
	pop de	209
	jr next block	24 14
"Subtract" (resta)	cp 1	254 1
	jr nz, next block	32 10
	push hl	229
	ex de,hl	235
	ld de, 2016	17 224 7
	and a	167
	sbc hl,de	237 82
	ex de,hl	235
	pop hl	225
"Next Block" (siguiente bloque)	ld a,c	121
	and 63	230 63
	cp O	254 O
	jr nz, add	32 6
	ld a,7	62 7
	add a,h	132
	ld h,a	103
	jr next line	24 187

"Add" (suma)	cp 1	254 1
	jr nz, next line	32 183
	ld a,7	62 7
	add a,d	130
	ld d,a	87
	ld a,c	121
	cp 1	254 1
	jr nz, next line	32 174
	ret	201

Funcionamiento:

El registro dual *hl* se carga con la dirección del "archivo de información gráfica" y el registro dual *de* se carga con la dirección del primer octeto de la segunda línea de información gráfica. El registro *c* se carga con el número de líneas que hay en la información gráfica. El registro *b* se carga con el número de octetos que hay en una línea, para usarse como contador.

El acumulador se carga con el octeto direccionado por *de*. Este se carga luego en la dirección expresada en *hl*. El acumulador se carga con el contenido del registro *c*. Si éste contiene el valor 2, el registro *de* apunta a la línea inferior de la pantalla, y de este modo se carga con cero. Los registros *hl* y *de* se incrementan para apuntar a los siguientes octetos. El contador en el registro *b* se decrementa y si no contiene un cero la rutina vuelve al lazo o bucle encabezado por "Copy Byte".

El número 224 se suma a los registros *hl* y *de*, de modo que apunten a la línea siguiente de información gráfica. El contador de líneas, que está en el registro *c*, se decrementa. Si el valor en *c* no es un múltiplo de ocho se ejecuta un salto a "subtract". 2016 se resta de *hl*, y se ejecuta un salto a "Next Block". Esto tiene como fin hacer que *hl* apunte al siguiente conjunto de ocho líneas.

Durante el lazo "subtract", si el valor (c-1) no es un multiplo de ocho se ejecuta un salto a "Next Block", de lo contrario 2016 se restará del registro *de*, de modo que apunte al siguiente conjunto de ocho líneas. En el lazo "Next Block", si *c* es un multiplo de 64, 1792 se suma a *hl*,y se produce un salto a "Next Line", de forma que *hl* apunte al siguiente bloque de 64 líneas. En el lazo "add", si (c-1) es multiplo de 64, 1792 se suma al *de* de modo que el registro *de* apunte al siguiente bloque de 64 líneas. Si *c* no contiene un 1 la rutina salta a "Next Line", de lo contrario la rutina retorna al Basic.

"SROLL" en Sentido Descendente por un Elemento de Imagen

Longitud: 90

Número de variables: 0

Prueba de la suma: 9862

Operación:

Esta rutina hace "scroll" del contenido del "archivo de información gráfica", en sentido descendente por un elemento de imagen.

"Llamada" (Call)

RAND USR dirección

"Comprobación de Errores" (Error Checks)

Ninguna

Comentarios

Ninguno

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, 22527	33 255 87
	ld de, 22271	17 255 86
	ld c, 192	14 192
"Next Line" (siguiente línea)	ld b, 32	6 32
"Copy Byte" (copiar octeto)	ld a, (de)	26
	ld (hl),a	119
	ld a,c	121
	cp 2	254 2
	jr nz, next byte	32 2
	sub a	151
	ld (de),a	18
"Next Byte" (siguiente octeto)	dec de	27
	dec hl	43
	djnz, copy byte	16 243
	push de	213
	ld de, 224	17 224 O
	and a	167
	sbc hl,de	237 82
	ex (sp), hl	227
	and a	167
	sbc hl,de	237 82
	ex de,hl	235
	pop hl	225
	dec c	13
	ld a,c	121
	and 7	230 7
	cp O	254 O
	jr nz, add	32 8
	push de	213
	ld de, 2016	17 224 7
	add hl,de	25
	pop de	209
	jr next block	24 11
"Add" (suma)	cp 1	254 1
	jr nz, next block	32 7

```
                                    push hl              229
                                    ld hl, 2016          33 224 7
                                    add hl,de            25
                                    ex de,hl             235
                                    pop hl               225
"Next Block" (siguiente bloque)     ld a,c               121
                                    and 63               230 63
                                    cp O                 254 O
                                    jr nz, subtract      32 6
                                    ld a,h               124
                                    sub 7                214 7
                                    ld h,a               103
                                    jr next line         24 188
"Subtract" (resta)                  cp 1                 254 1
                                    jr nz, next line     32 184
                                    ld a,d               122
                                    sub 7                214 7
                                    ld d,a               87
                                    ld a,c               121
                                    cp 1                 254 1
                                    jr nz, next line     32 175
                                    ret                  201
```

Funcionamiento:

El registro dual *hl* se carga con la dirección del último octeto del archivo de información gráfica, y el registro *de* se carga con la dirección del octeto que está una línea por encima del último octeto. El registro *c* se carga con el número de líneas de información gráfica. El registro *b* se carga con el número de octetos que hay en una línea, para usarse como contador.

El acumulador se carga con el octeto direccionado por el registro *de.* Esto es cargado luego en la dirección contenida en *hl.* El acumulador se carga con el contenido del registro *c.* Si este contiene el valor dos, el registro *de* apunta a la línea superior de la pantalla, de este modo se carga con el valor 0. El registro *de* y el *hl* son posteriormente decrementados para apuntar a los siguientes octetos. El contador en el registro *b* se decrementa, y si este no contiene un cero la rutina se dirige al lazo "Copy Byte".

224 se resta de los registros *hl* y *de,* de forma que apunten a la siguiente línea de información gráfica. El contador de línea en el registro *c,* se decrementa. Si el valor en *c* no es un multiplo de ocho, salta a "add". 2016 se suma luego con *hl,* y se salta a "Next Block". Esto es, apuntar *hl* al siguiente bloque de ocho líneas.

En "Add", si el valor (c-1) no es un multiplo de ocho, se salta a "Next Block". 2016 se suma a *de* de modo que apunte al siguiente conjunto de ocho líneas. En "Next Block", si *c* es multiplo de 64, 1792 se sustrae de *hl* de forma que *hl* apunte al siguiente bloque de 64 líneas, y se ejecuta un salto en dirección a "Next Line". Al sustraer, si (c-1) es un multiplo de 64, 1792 se sustrae

del registro *de,* de forma que *de* apunte al siguiente bloque de 64 líneas. Si *c* no contiene un uno, la rutina salta a "Next Line", de lo contrario lá rutina vuelve al Basic.

6. RUTINAS DE INFORMACION GRAFICA

Combinación de imágenes

Longitud: 21
Número de variables: 1
Prueba de la suma: 1709

Operación:

Esta rutina combina una imagen almacenada en la RAM (usando la rutina "Copy" en otra parte de este libro), con la información gráfica actual de la pantalla (la que hubiere en ese momento). Las cualidades se mantienen, no cambian.

Variables:

Nombre	*Longitud*	*Posición*	*Comentario*
Almacén de imagen	2	23296	Dirección en la RAM de la imagen almacenada.

"Llamada" (Call)

RAND USR direción

"Comprobación de errores" (Error Checks)

Ninguna

Comentarios

Para combinar imágenes, la rutina deberá usarse como se da en el listado. Sin embargo se pueden producir efectos de interés si reemplazamos la instrucción "or (hl) 182" por "xor (hl) 174" o por "and (hl) 166".

Listado del código máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, 16384	33 0 64
	ld de, (23296)	237 91 0 91
	ld bc, 6144	1 0 24
"Next Byte" (siguiente octeto)	ld a, (de)	26
	or (hl)	182
	ld (hl),a	119
	inc hl	35
	inc de	19
	dec bc	11
	ld a,b	120
	or c	177
	jr nz, next byte	32 246
	ret	201

Funcionamiento:

El registro dual *hl* se carga con la dirección del archivo de información gráfica y el registro dual *de* se carga con la longitud de la información gráfica, de forma que se pueda usar como contador.

El acumulador se carga con el octeto contenido en la dirección almacenada en *de* y a esto y al siguiente octeto del archivo de información gráfica se les aplica el operador lógico "o" ("or", ver glosario en capítulo 4). El valor resultante es cargado de nuevo en la información gráfica.

El registro *hl* y el *de* se mueven sobre la siguiente posición, y el contador se decrementa. Si el contador no es cero la rutina vuelve atrás para repetir el proceso sobre el siguiente octeto.

Inversión de pantalla

Longitud: 18
Número de variables: 0
Prueba de la suma: 1613

Operación:

Invierte todo el archivo de información gráfica, es decir, donde hay un punto activo (encendido/on) lo pone en estado desactivado y al revés.

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Ninguna

Comentarios:

Esta rutina puede usarse en programas de juegos para producir un efecto de explosión. El efecto se aumenta si llamamos a la rutina varias veces, añadiéndole alguna forma de sonido.

Listado del código máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, 16384	33 0 64
	ld bc, 6144	1 0 24
	ld d, 255	22 255
"Next Byte" (siguiente octeto)	ld a,d	122
	sub (hl)	150
	ld (hl),a	119
	inc hl	35
	dec bc	11
	ld a,b	120

	or c	177
	jr nz, next byte	32 247
	ret	201

Funcionamiento:

El registro dual *hl* se carga con la dirección del archivo de información gráfica y el registro *bc* se carga con su longitud. El registro *d* se emplaza a 255. Cada vez que la rutina vuelve al principio del lazo "Next Byte", el acumulador se carga desde *d*. Se usa este método en vez del "ld , 255" debido a que la instrucción "ld a, d" emplea la mitad de tiempo que la instrucción "ld a, 255". El valor del octeto almacenado en *hl* se sustrae del acumulador, y el resultado se carga de nuevo en el mismo octeto, esto es, invirtiéndolo.

El registro *hl* se incrementa para apuntar al siguiente octeto, y el contador *bc* se decrementa. Si el contador no es cero, la rutina vuelve atrás hasta "Next Byte". Si el contador es cero, la rutina vuelve al Basic.

Inversión de un Carácter Verticalmente

Longitud: 20

Número de variables: 1

Prueba de la suma: 1757

Operación

Esta rutina invierte un carácter verticalmente, o lo que es igual, por ejemplo una flecha apuntando hacia arriba se transforma en la misma flecha pero apuntando hacia abajo y al revés.

Variables:

Nombre	*Longitud*	*Posición*	*Comentario*
Comienzo del carácter	2	23296	Dirección de los datos del carácter en la RAM

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Ninguna

Comentarios

Esta rutina es útil en juegos donde los símbolos puedan cambiar de dirección sin usar más de un carácter.

Listado del código máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, (23296)	42 0 91
	ld d,h	84

	ld e,l	93
	ld b,8	6 8
"Next Byte" (siguiente octeto)	ld a, (hl)	126
	inc hl	35
	push af	245
	djnz next byte	16 251
	ld b,8	6 8
"Replace" (reemplazar)	pop af	241
	ld (de),a	18
	inc de	19
	djnz replace	16 251
	ret	201

Funcionamiento

El registro dual *hl* se carga con la dirección del dato del carácter en la RAM. Luego se copia en el registro *de.* AL registro *b* se le da el valor 8, para usarse como contador.

Por cada octeto, se carga el acumulador con ese valor, *hl* se incrementa para apuntar al siguiente octeto, y el acumulador es "empujado (por medio de "Push") al área de conservación de registros. El contador se decrementa, y si no es cero su contenido, la rutina retrocede para repetir el proceso para el siguiente octeto. El registro *b* es recargado con 8 para usarlo de nuevo como contador.

Por cada octeto, el acumulador es "hecho saltar" desde el área de conservación de registros, e introducido (Poke) en la dirección almacenada en el registro *de*. El registro *de* se decrementa para apuntar al siguiente octeto y el contador se decrementa. Si éste no es cero la rutina retrocede a "Replace".

Luego se vuelve al Basic.

Inversión del Carácter Horizontalemente

Longitud: 19
Número de variables: 1
Prueba de la suma: 1621

Operación

Esta rutina invierte un carácter horizontalmente —ejemplo, una flecha apuntando hacia la derecha se transforma en otra flecha apuntando a la izquierda y al revés.

Variables:

Nombre	*Longitud*	*Posición*	*Comentario*
Comienzo del carácter	2	23296	Dirección de los datos del carácter en la RAM.

"llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Ninguna

Comentarios

Ninguno

Listado del código máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, (23296)	42 0 91
	ld a, 8	62 8
"Next Byte" (siguiente octeto)	ld b, 8	6 8
"Next Pixel" (siguiente elemento de imagen)	rr (hl)	203 30
	rl c	203 17
	djnz next pixel	16 250
	ld (hl),c	113
	inc hl	35
	dec a	61
	jr nz, next byte	32 243
	ret	201

Funcionamiento

El registro dual *hl* se carga con la dirección del dato del carácter en la RAM, el acumulador se carga con el número de octetos que serán invertidos. El registro *b* se carga con el número de "bits" que hay en cada octeto, para usarse como contador.

El octeto que hay en la dirección expresada por *hl* se hace rotar a la derecha de forma que el bit más a la derecha sea copiado en el indicador de acarreo. El registro *c* se rota a izquierdas de forma que el indicador de acarreo sea copiado en el bit más a la derecha. El contador que hay en el registro *b* se decrementa. Si el contador no es cero, se ejecuta un salto atrás a "Next Pixel" (siguiente elemento de imagen). El octeto invertido, que está almacenado en el registro *c,* es cargado en la dirección de la que originalmente partió.

El registro *hl* se decrementa para apuntar al siguiente octeto y el acumulador se decrementa. Si el acumulador se decrementa, ejecuta un salto atrás a "Next Byte".

Luego se vuelve al Basic.

Rotar un Cafacter en el Sentido de las Manecillas del Reloj

Longitud: 42

Número de varibles: 1

Prueba de la suma: 3876

Operación

Esta operación rota un carácter 90° en el sentido de las agujas del reloj. Por ejemplo, una flecha apuntando hacia arriba se convierte en una flecha apuntando hacia la derecha.

Variables:

Nombre	*Longitud*	*Posición*	*Comentario*
Comienzo del carácter	2	23296	Dirección de los datos del carácter en la RAM.

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Ninguna

Comentarios

Esta rutina es útil en juegos y en aplicaciones más serias. Por ejemplo, etiquetado de gráficos.

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, (23296)	42 0 91
	ld e, 128	30 128
"Next Bit" (siguiente bit)	push hl	229
	ld c, O	14 0
	ld b,1	6 1
"Next Byte" (siguiente octeto)	ld a, e	123
	and (hl)	166
	cp O	254 0
	jr z, not set	40 3
	ld a,c	121
	add a,b	128
	ld c,a	79
"Not set" (no emplazado)	sla b	203 32
	inc hl	35
	jr nc, next byte	48 242
	pop hl	225
	push bc	197
	srl e	203 59
	jr nc, next bit	48 231
	ld de,7	17 7 0
	add hl,de	25
	ld b,8	6 8

"Replace" (reemplazar)	pop de	209
	ld (hl),e	115
	dec hl	43
	djnz replace	16 251
	ret	201

Funcionamiento

Cada carácter consiste en un grupo de 8 × 8 elementos de imagen, cada uno de los cuales puede activarse (poner a 1) o desactivarse (poner a 0). Considerar cualquier bit B_2 de un octeto B_1 de la figura B1. Los datos contenidos en la posición (B_2, B_1) en la matriz será:

$$\begin{pmatrix} N_1 & N_3 \\ N_2 & N_4 \end{pmatrix}$$

donde:

N_1 = al octeto en el que se deberá insertar el elemento de imagen (B_2, B_1) después de una rotación.

N_2 = el bit en N_1 en el que se debe insertar.

N_3 = El valor del bit que se representa en ese momento.

N_4 = El valor del bit N_2.

Cada octeto del carácter rotado deberá ser construído de uno en uno, mediante la suma de todos los valores de todos los bits N_2 que estarán en el nuevo octeto.

El registro *hl* se carga con la dirección del primer octeto del carácter en la RAM. El registro *e* se carga con el valor del octeto que tenga el bit 7 activado y los bits 0 al 6 desactivados, por ejemplo 128. El registro *hl* se guarda en el área de conservación de registros (Stack). El registro *c*, al que se le deben sumar sus datos (bits o también pueden ser octetos) dándole el nuevo valor del octeto que está siendo construído, se carga con cero. El registro *b* se carga con el valor del octeto que tenga el bit 0 activado y los bits 1 al 7 desactivados, por ejemplo el 1.

El siguiente cuadro concuerda con la página 76.

1 128	2 64	3 32	4 16	5 8	6 4	7 2	8 1	1
0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	
1 128	2 64	3 32	4 16	5 8	6 4	7 2	8 1	2
1 2	1 2	1 2	1 2	1 2	1 2	1 2	1 2	
1 128	2 64	3 32	4 16	5 8	6 4	7 2	8 1	3
2 4	2 4	2 4	2 4	2 4	2 4	2 4	2 4	

1 128	2 64	3 32	4 16	5 8	6 4	7 2	8 1	4
3 8	3 8	3 8	3 8	3 8	3 8	3 8	3 8	Octeto (B_1)
1 128	2 64	3 32	4 16	5 8	6 4	7 2	8 1	5
4 16	4 16	4 16	4 16	4 16	4 16	4 16	4 16	
1 128	2 64	3 32	4 16	5 8	6 4	7 2	8 1	6
5 32	5 32	5 32	5 32	5 32	5 32	5 32	5 32	
1 128	2 64	3 32	4 16	5 8	6 4	7 2	8 1	7
6 64	6 64	6 64	6 64	6 64	6 64	6 64	6 64	
1 128	2 64	3 32	4 16	5 8	6 4	7 2	8 1	8
7 128	7 128	7 128	7 128	7 128	7 128	7 128	7 128	
7	6	5	4	3	2	1	0	

Bit (B_2)

Figura B1. *Clave para la rutina de rotación de caracteres.*

El acumulador se carga con el contenido del registro *e*, (N_3). A todo esto juntamente con el octeto cuya dirección está almacenada en el *hl* se le aplica el operador lógico "y" (And). Si el resultado es un cero se ejecuta un salto a "Not Set", así como el elemento de imagen direccionado por el registro *e* y el registro *hl* es desactivado. Si se activa, el acumulador se carga con el valor presente del octeto, (N_1). El registro *b,* (N_4), se suma al acumulador y éste es cargado en *c*. Luego el registro *b* se ajusta para apuntar al siguiente bit de N_1. El registro *hl* se incrementa para apuntar al siguiente octeto, (B_1). Si el octeto N_1 no está completo, la rutina vuelve a "Next Byte".

El registro *hl* se recupera desde el área de conservación de registros para apuntar de nuevo al primer octeto del carácter. El registro *bc* se guarda en el área de conservación de registros, conteniendo el valor del último octeto que va a ser completado en el registro *c*. El registro *e* se ajusta a la dirección del siguiente bit de cada octeto. Si la rotación no está completa se vuelve, mediante una instrucción de "salto" (Jump), a "Next Bit".

El registro dual *de* se carga con un 7, y esto se suma con *hl* de forma que *hl* apunte al último octeto de datos. Para cada octeto, el nuevo valor se copia en *e,* y se carga en *hl.* El registro *hl* se decrementa para apuntar al siguiente octeto, y el contador en el registro *b* se decrementa. Si el contador no contiene un cero, se ejecuta un salto a "Replace".

La rutina vuelve al Basic.

Cambio de Cualidad

Longitud: 21
Número de variables: 2
Prueba de la suma: 1952

Operación

Esta rutina altera las cualidades de todos los caracteres sobre la pantalla de una forma específica. Por ejemplo, el color de la "tinta" (Ink) puede cambiarse, así como activar el efecto "Flash" (destello) sobre toda la pantalla, etc.

Variables:

Nombre	*Longitud*	*Posición*	*Comentario*
Datos a guardar	1	23296	Bits de la cualidad que no serán alterados.
Nuevos datos	1	23297	Nuevos bits que se insertarán en la cualidad.

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Ninguna

Comentarios

Los bits de las cualidades tomados individualmente, de cada carácter, se pueden cambiar usando las instrucciones en código máquina "And" ("y") y "Or" ("o").

Listado del código máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, 22528	33 0 88
	ld bc, 768	1 0 3
	ld de, (23296)	237 91 0 91
"Next Byte" (siguiente octeto)	ld a, (hl)	126
	and e	163
	or d	178
	ld (hl), a	119
	inc hl	35
	dec bc	11
	ld a,b	120
	or c	177
	jr nz, next byte	32 246
	ret	201

Funcionamiento:

El registro dual *hl* se carga con la dirección del área de cualidades, el registro dual *bc* se carga con el número de caracteres que hay en la información gráfica. El registro *d* se carga con el valor de "New Data" (Nuevos datos), y el registro *e* se carga con "Data Saved" (Datos guardados).

El registro acumulador se carga con el octeto direccionado por *hl*, y los bits se ajustan de acuerdo a los valores de los registros *d* y *e*. El resultado es cargado otra vez en *hl*. El registro *hl* se incrementa para que apunte al siguiente octeto, y el contador que hay en *bc* se decrementa. Si el registro *bc* no contiene un cero, la rutina ejecuta el lazo que comienza en "Siguiente Octeto".

La rutina vuelve al Basic.

Intercambio de Cualidad

Longitud: 22
Número de variables: 2
Prueba de la suma: 1825

Operación

Esta rutina busca en el "area de cualidades" un cierto valor y reemplaza cada valor por otro.

Variables:

Nombre	*Longitud*	*Posición*	*Comentario*
"Antiguo valor"	1	23296	Valor del octeto que será reemplazado.
"Nuevo valor"	1	23297	Nuevo valor del octeto reemplazado.

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Ninguna

Comentarios

Esta rutina es útil para destacar áreas de texto y caracteres de gráficos.

Listado del código máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, 22528	33 0 88
	ld bc, 768	1 0 3
	ld de, (23296)	237 91 0 91

"Next Byte" (siguiente octeto)	ld a, (hl)	126
	cp e	187
	jr nz, no change	32 1
	ld (hl), d	114
"No Change" (no cambiar)	inc hl	35
	dec bc	11
	ld a,b	120
	or c	177
	jr nz, next byte	32 245
	ret	201

Funcionamiento

El registro dual *hl* se carga con la dirección del área de cualidades, y el registro *bc* se carga con el número de caracteres que hay en la pantalla. El registro *e* se carga con el "antiguo valor", el registro *d* se carga con el "nuevo valor".

El acumulador se carga con el octeto direccionado por el registro dual *hl*. Si el acumulador contiene el valor del registro *e* el octeto direccionado por *hl* se carga con el contenido del registro *d*. El registro *hl* se incrementa luego para que apunte al siguiente octeto, y el contador que hay en *bc* se decrementa. Si el registro *bc* no contiene un cero, se ejecuta un salto a "Next Byte" (octeto siguiente).

La rutina vuelve después al Basic.

"Rellenando Zonas"

Longitud: 263
Número de variables: 2
Prueba de la suma: 26647

Operación

Esta rutina "sombrea" una zona de la pantalla rodeada por una línea de elementos de imagen sobre el borde de la pantalla.

Variables:

Nombre	*Longitud*	*Posición*	*Comentario*
Coordenada *x*	1	23296	Posición de arranque en la coordenada de las "x".
Coordenada *y*	1	23297	Posición de arranque en la coordenada de las "y".

"Llamada" (Call)

RAND USR dirección

Comprobación de errores" (Control Checks)

Si la coordenada "y" es más de 175, o "POINT" (x, y) = 1 la rutina devuelve el control al Basic inmediatamente.

Comentarios

Esta rutina no es relocalizable, la dirección de comienzo es 31955. Para transferir esta rutina a otra dirección aconsejemos usar el método dado para la rutina "RENUMERAR" (Renumber). Si 31955 se usa como dirección de comienzo de la rutina "RENUMERAR", entonces ambas pueden estar alojadas en la RAM simultáneamente.

Cuando sombreamos configuraciones de zonas muy irregulares, necesitamos una gran cantidad de espacio disponible en RAM. Si no disponemos de este espacio la rutina puede dejar de funcionar.

Listado del código máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, (23296)	42 0 91
	ld a,h	124
	cp 176	254 176
	ret nc	208
	call subroutine	205 143* 125*
	and (hl)	166
	cp O	254 O
	ret nz	192
	ld bc, 65535	1 255 255
	push bc	197
"Right" (derecha)	ld hl, (23296)	42 0 91
	call subroutine	205 143* 125*
	and (hl)	166
	cp O	254 O
	jr nz, left	32 9
	ld hl, (23296)	42 0 91
	inc l	44
	ld (23296),hl	34 0 91
	jr nz, right	32 236
"Left" (izquierda)	ld de, O	17 0 0
	ld hl, (23296)	42 0 91
	dec l	45
	ld (23296),hl	34 0 91
"Plot" (trazar)	ld hl, (23296)	42 0 91
	push hl	229
	call subroutine	205 143* 125*
	or (hl)	182
	ld (hl),a	119
	pop hl	225

	ld a,h	124
	cp 175	254 175
	jr z, down	40 44
	ld a,e	123
	cp O	254 O
	jr nz, reset	32 16
	inc h	36
	call subroutine	205 143* 125*
	and (hl)	166
	cp O	254 O
	jr nz, reset	32 7
	ld hl, (23296)	42 0 91
	inc h	36
	push hl	229
	ld e,l	30 1
"Reset" (desactivar)	ld hl, (23296)	42 0 91
	ld a,e	123
	cp 1	254 1
	jr nz, down	32 15
	inc h	36
	call subroutine	205 143* 125*
	and (hl)	166
	cp O	254 O
	jr z, down	40 6
	ld e, O	30 0
	jr down	24 2
"Long jump" (salto largo)	jr right	24 167
"Down" (sentido descendente)	ld hl, (23296)	42 0 91
	ld a,h	124
	cp O	254 O
	jr z, next pixel	40 40
	ld a,d	122
	cp O	254 O
	jr nz, restore	32 16
	dec h	37
	call subroutine	205 143* 125*
	and (hl)	166
	cp O	254 O
	jr nz, restore	32 7
	ld hl, (23296)	42 0 91
	dec h	37
	push hl	229
	ld d,1	22 1
"Restore" (recargar)	ld a,d	122
	cp 1	254 1
	jr nz, next pixel	32 14

	ld hl, (23296)	42 0 91
	dec h	37
	call subroutine	205 143* 125*
	and (hl)	166
	cp O	254 O
	jr z, next pixel	40 2
	ld d, O	22 O
"Next Pixel" (Siguiente elemento de imagen)	ld hl, (23296)	42 O 91
	ld a,l	125
	cp O	254 O
	jr z, retrieve	40 12
	dec l	45
	ld (23296),hl	34 0 91
	call subroutine	205 143* 125*
	and (hl)	166
	cp O	254 O
	jr z, plot	40 129
"Retrieve" (recuperar o recoger)	pop hl	225
	ld (23296),hl	34 0 91
	ld a, 255	62 255
	cp h	188
	jr nz, long jump	32 177
	cp 1	189
	jr nz, long jump	32 174
	ret	201
"Subroutine' (Subrutina)	push bc	197
	push de	213
	ld a, 175	62 175
	sub h	148
	ld h,a	103
	push hl	229
	and 7	230 7
	add a, 64	198 64
	ld c,a	79
	ld a,h	124
	rra	203 31
	rra	203 31
	rra	203 31
	and 31	230 31
	ld b, a	71
	and 24	230 24
	ld d,a	87
	ld a,h	124
	and 192	230 192
	ld e,a	95
	ld h,c	97

	ld a,l	125
	rra	203 31
	rra	203 31
	rra	203 31
	and 31	230 31
	ld l,a	111
	ld a,e	123
	add a,b	128
	sub d	146
	ld e,a	95
	ld d, O	22 O
	push hl	229
	push de	213
	pop hl	225
	add hl,hl	41
	add hl,hl	41
	add hl,hl	41
	add hl,hl	41
	add hl,hl	41
	pop de	209
	add hl,de	25
	pop de	209
	ld a,e	123
	and 7	230 7
	ld b, a	71
	ld a, 8	62 8
	sub b	144
	ld b,a	71
	ld a,l	62 1
"Rotate" (rotar)	add a,a	135
	djnz rotate	16 253
	rra	203 31
	pop de	209
	pop bc	193
	ret	201

Funcionamiento

Esta rutina traza las líneas horizontales de elementos de imagen adyacentes llamados "RUNS", dentro de áreas rodeadas por elementos de imagen activados (iluminados). Tenemos constancia de cada "RUN" por medio del almacenaje en el área de conservación de registros de las coordenadas del elemento de imagen más a la derecha del "RUN". Comenzando desde unas coordenadas específicas, la rutina rellena cada posición con cada elemento "RUN", tomando nota de las posiciones arriba o abajo que cualquier "RUN" deje sin rellenar. Cuando un "RUN" se completa, se recoge el último conjunto de coordenadas que fue anotado y el correspondiente "RUN" se rellena en él. El proceso se repite hasta que no haya más "RUNS" sin rellenar.

La figura B2. nos ilustra esta técnica. Los cuadros representan elementos de imagen iluminados; las marcas con "x" señalan la posición de arranque dentro del área que será sombreada, y el * señala los elementos de imagen más a la derecha de los "RUNS".

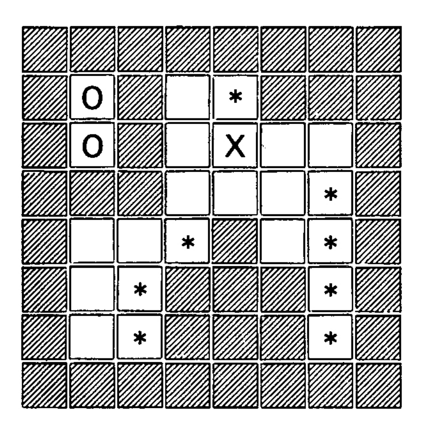


Figura B2. *Ejemplo de la técnica empleada para rellenar una zona. Los cuadros en gris están iluminados ya y definen el área que será sombreada. "X" es la posición de arranque, * son los comienzos de los "RUNS" y los símbolos "O" permanecen sin sombrear.*

La rutina sombrea la línea horizontal que contiene la posición de comienzo y guarda en el área de conservación de regitros las posiciones de comienzo de los "RUNS" en las líneas inmediatas por arriba y por abajo. Hace que a continuación se sombree la línea de arriba y luego la de abajo, si bien en este último caso hay que considerar que dos "RUNS" más comienzan en la línea siguiente inferior y así sucesivamente. Cualquier posición dentro del área que sera sombreada podrá seleccionarse como la posición de comienzo. Pero hay que recalcar, que dos elementos de imagen señalados con ceros se dejan sin tocar debido a que están separados del área que está siendo sombreada.

El registro *h* se carga con la coordenada "y" especificada, el registro *l* se carga con la coordenada "x". Si el valor de la coordenada "y" es más que 175 la rutina vuelve al Basic. La "subrutina" es llamada por medio de la devolución de la dirección, en memoria, del bit correspondiente en las dos coordenadas (x, y). Si este bit está "activado" (On) la rutina vuelve al Basic.

El número 65535 es "empujado" (Push) al área de conservación de registros (Stack), para señalar el primer valor guardado. Posteriormente en la rutina, si un número es recuperado desde el área de conservación de registros, será usado éste como una pareja de coordenadas. Sin embargo, si el número resultante es 65535, se vuelve al Basic ya que la rutina ha acabado.

El registro *h* se carga con la coordenada "y", y el registro *l* se carga con la coordenada "x". La forma en que se llama a la "subrutina" es, devolviendo en el registro dual *hl* la dirección del bit (x, y). Si este bit está "activado" se eje-

cuta un salto a "left". De lo contrario, la coordenada "x" se verá incrementada y ejecutará un lazo que comienza en "right", si "x" no es igual a 256.

En "left", el registro dual *de* es emplazado a cero. Los registros *d* y *e* van a usarse como señalizadores, *d* para abajo y *e* para arriba, La coordenada "x" se decrementa. Se llama a la subrutina y el punto definido por (x, y) es trazado a continuación. Si la coordenada "y" es 175, la rutina salta a "down". Si el "up flag" (Indicador de arriba o sentido ascendente) está activado (es decir, contine un 1), se ejecuta un salto a "reset". Si el bit (x,y + 1) está "desactivado" los valores de x y de y + 1 serán guardados en el área de conservación de registros, y el indicador "up flag" se emplazará a 1 (se activa, toma el valor 1).

En "reset", si el indicador "up flag" está emplazado a cero, se ejecutará un salto a "down". Si el bit (x,y + 1) está activado (On) el indicador de "up flag" se emplaza a cero. En "down", si la coordenada "y" es cero se ejecuta un salto a "Next Pixel". Si el indicador "down flag" tiene un uno se ejecutará un salto a "restore". Si el bit (x, y—1) está desactivado (Off) los valores de x y de y—1 serán guardados en el área de conservación de registros, y por consiguiente el indicador de "down flag" se emplaza a uno.

En "restore", si el indicador de "down flag" está a cero se ejecuta un salto a "Next Pixel". Si el bit (x, y—1) está activo, el indicador de "down flag" está emplazado a cero. En "Next Pixel", si la coordenada es cero la rutina salta a "retrieve. La coordenda "x" se decrementa, y si el nuevo bit (x,y) está desactivado, se ejecuta un salto a "plot". En "retrieve", las coordenadas "x" e "y" son sacadas del área de conservación de registros. Si tanto "x" como "y" tienen un valor de 255, la rutina vuelve al Basic ya que la zona se ha rellenado completamente. De lo contrario, la rutina vuelve al "right".

La subrutina ha de calcular la dirección del bit (x,y) en memoria. En Basic esta dirección podría ser:

$$16384 + \text{INT}(Z/8) + 256 \times (Z - 8 \times \text{INT}(Z/8)) + 32 \times (64 \times \text{INT}(Z/64) + \text{INT}(Z/8) - 8 \times \text{INT}(Z/64))$$

donde $Z = 175 - y$

Los registros duales *bc* y *de* son guardados en el área de conservación de registros. Los cinco bits situados a la izquierda del acumulador se emplazan a cero, después se le suma el número 64. El resultado se transfiere la registro *c*. Cuando esto se multiplica por 256 nos dará: 16384 + 256 × (Z—8 × INT(Z/8)). El acumulador se carga con Z, éste se divide por 8, el resultado se trasfiere al registro *b*. Este resultado es INT(Z/8). Emplazando los tres bits más a la derecha a cero, nos produce el valor siguiente: 8 × INT (Z/64), y esto será cargado en el registro *d*.

El acumulador se carga con Z, y los seis bits más a la derecha se emplazan a cero, produciendo el valor 64 × INT (Z/64). Esto será cargado en el registro *e*. El valor en el registro *c* se transfiere al registro *h*. El acumulador se carga con la coordenada "x", se divide por ocho, y el resultado se transfiere a *l*.

El acumulador se carga luego con el valor en *e,* al cual se le suma el valor en *b*. El valor en *d* se sustraé de lo anterior y el resultado se carga en *de*. El

registro dual *hl* se guarda en el área de conservación de registros y cargado luego con el valor que hay en *de*. Esto se multiplica por 32, *de* se recoge del área de conservación de registros y se suma a *hl*. Ahora *hl* contiene la dirección del bit (x,y).

El acumulador se carga con el valor original de *x*. Emplazando los cinco bits, situados a la izquierda, a cero, produce el valor x — 8 × INT (x/8). El registro *b* se carga luego con 8 menos el valor del acumulador; para usarse como un contador. El acumulador se emplaza a uno, y éste es multiplicado por dos las (b-1) veces.

Llegados a este punto, un simple bit deberá estar activo en el acumulador, lo que corresponde al bit (x,y) direccionado por *hl*. Los registros *de* y *bc* son recogidos del área de conservación de registros, y luego la subrutina retorna a la rutina principal.

Tablas de figuras

Longitud: 196
Número de variables: 2
Prueba de la suma: 20278

Operación

Esta rutina traza una figura de cualquier tamaño sobre la pantalla

Variables

Nombre	*Longitud*	*Posición*	*Comentario*
Arranque en "x"	1	23296	Coordenada "x" del primer elemento de imagen.
Arranque en "y"	1	23297	Coordenada "y" del primer elemento de imagen.

"Llamada" (Call)

RAND USR dirección

Comprobación de errores (Error Checks)

En los casos en que A$ no exista, tenga longitud cero, o no contenga información de cualquier figura, la rutina vuelve al Basic inmediatamente. Esto también ocurre si el arranque en "y" es más que 175.

Comentarios

Es un método útil de almacenar figuras en memoria que van a trazarse a la velocidad del barrido inherente a la pantalla.

El método para usar esta rutina es:

I) LET A$ = "Información de la figura"
II) POKE 23296, coordenada X del primer elemento de imagen.
III) POKE 23297, coordenada Y del primer elemento de imagen.
IV) RAND USR dirección.

La información de la figura es una cadena de caracteres, que tendrá las siguientes características:

"O" Trazar punto
"5" Decrece la coordenada "X"
"6" Decrece la coordenadaa "Y"
"7" Incrementa la coordenada "Y"
"8" Incrementa la coordenada "X"

Ignorará cualquier otro carácter.

La rutina incluye una propiedad de "circulación". Por ejemplo, si la coordenada X desaparece en la parte izquierda de la pantalla, aparecerá en la parte derecha, etc.

Para hacer que la rutina emplee una cadena distinta a A$, cambiar el número 65 que lleva asterisco en la columna "números a introducir" por el código correspondiente al cafacter en mayúscula que queramos que figure como nombre de la cadena.

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, (23627)	42 75 92
"Next Variable" (siguiente variable)	ld a, (hl)	126
	cp 128	254 128
	ret z	200
	bit 7,a	203 127
	jr nz, for next	32 23
	cp 96	254 96
	jr nc, number	48 11
	cp 65	254 65*
	jr z, found	40 35
"String" (Cadena)	inc hl	35
	ld e, (hl)	94
	inc hl	35
	ld d, (hl)	86
"Add" (suma)	add hl,de	25
	jr increase	24 5
"Number" (número)	inc hl	35
	inc hl	35
	inc hl	35
	inc hl	35

	inc hl	35
"Increase" (incrementar)	inc hl	35
	jr next variable	24 225
"For Next" (Con... Otra)	cp 224	254 224
	jr c, next bit	56 5
	ld de, 18	17 18 0
	jr add	24 236
"Next Bit" (siguiente bit)	bit 5,a	203 111
	jr z, string	40 228
"Next Byte" (siguiente octeto)	inc hl	35
	bit 7,(hl)	203 126
	jr z, next byte	40 251
	jr number	24 228
"Found" (encontrar)	inc hl	35
	ld c, (hl)	78
	inc hl	35
	ld b, (hl)	70
	inc hl	35
	ex de,hl	235
	ld a, (23297)	58 1 91
	cp 176	254 176
	ret nc	208
"Again" (de nuevo)	ld hl, (23296)	42 0 91
	ld a,b	120
	or c	177
	ret z	200
	dec bc	11
	ld a, (de)	26
	inc de	19
	cp 48	254 48
	jr nz, not plot	32 78
	push bc	197
	push de	213
	ld a, 175	62 175
	sub h	148
	ld h,a	103
	push hl	229
	and 7	230 7
	add a,64	198 64
	ld c,a	79
	ld a,h	124
	rra	203 31
	rra	203 31
	rra	203 31
	and 31	230 31

	ld b,a	71
	and 24	230 24
	ld d,a	87
	ld a,h	124
	and 192	230 192
	ld e,a	95
	ld h,c	97
	ld a,l	125
	rra	203 31
	rra	203 31
	rra	203 31
	and 31	230 31
	ld l,a	111
	ld a,e	123
	add a,b	128
	sub d	146
	ld e,a	95
	ld d, O	22 O
	push hl	229
	push de	213
	pop hl	225
	add hl,hl	41
	add hl,hl	41
	add hl,hl	41
	add hl,hl	41
	add hl,hl	41
	pop de	209
	add hl,de	25
	pop de	209
	ld a,e	123
	and 7	230 7
	ld b,a	71
	ld a, 8	62 8
	sub b	144
	ld b,a	71
	ld a,1	62 1
"Rotate" (rotar)	add a,a	135
	djnz rotate	16 253
	rra	203 31
	pop de	209
	pop bc	193
	or (hl)	182
	ld (hl),a	119
"Here" (de nuevo aquí)	jr again	24 165
"Not Plot" (no trazar)	cp 53	254 53
	jr nz, down	32 1

	dec l	45
"Down" (abajo, sentido descendente)	cp 54	254 54
	jr nz, up	32 8
	dec h	37
	ld a,h	124
	cp 255	254 255
	jr nz, save	32 19
	ld h, 175	38 175
"Up" (arriba, sentido ascendente)	cp 55	254 55
	jr nz, right	32 8
	inc h	36
	ld a,h	124
	cp 176	254 176
	jr nz, save	32 7
	ld h, O	38 O
"Right" (derecha)	cp 56	254 56
	jr nz, save	32 1
	inc l	44
"Save" (guardar)	ld (23296),hl	34 0 91
	jr here	24 215

Funcionamiento

La dirección de la "cadena" A$ se encuentra usando una adaptación de la primera sección de la rutina llamada "INSTR$".

La longitud de la cadena se carga en el registro *bc,* y la dirección del primer carácter de A$ se carga en *de.* El acumulador es emplazado al valor de arranque en "y", y si éste es más de 175 la rutina vuelve al Basic. El registro *h* se carga con la coordenada "y", el registro *l* se carga con la coordenada "x". Si el valor del registro *bc* es cero, la rutina vuelve al Basic, debido a que hemos llegado al final de la cadena. El registro *bc* se decrementa, para indicar que un carácter más ha entrado en operación. El carácter siguiente se carga en el acumulador y el registro *de* se incrementa para apuntar al siguiente octeto. Si el acumulador no contiene un 48 se ejecuta un salto a "no plot". El punto (x,y) se traza usando la subrutina dada en la rutina "Rellenando zonas". La rutina salta luego hasta el encabezamiento "Again".

En "not plot", si el acumulador contiene un 53, la coordenada X se decrementa. En "down" si el acumulador no contiene un 54, se ejecuta un salto a "up". La coordenada Y se decrementa y si ésta contiene un —1 la coordenada Y es emplazada a 175.

En "up", si el acumulador no contiene un 55, se ejecuta un salto a "right". La coordenada Y se incrementa, y si ésta es 176, la coordenada Y es emplazada a 0. En "right", si el acumulador contiene un 56, la coordenada X se incrementa. En "save", las coordenadas X e Y son cargadas en memoria, y la rutina vuelve al comienzo del lazo encabezado por "here".

Agrandamiento y Copia de la Pantalla

Longitud: 335
Número de variables: 8
Prueba de la suma: 33663

Operación

Esta rutina copia una sección de la información gráfica de la pantalla en otra zona de la misma, agrandando la copia en los planos X o Y.

Variables

Nombre	*Longitud*	*Posición*	*Comentario*
Coordenada "Y" más alta	1	23296	Coordenada "Y" de la fila superior.
Coordenada "Y" más baja	1	23297	Coordenada "Y" de la fila inferior.
Coordenada "x" derecha	1	23298	Coordenada "x" de la columna más a la derecha.
Coordenada "x" izquierda	1	23299	Coordenada "x" de la columna más a la izquierda.
Escala horizontal	1	23300	Agrandamiento en el plano "x".
Escala vertical	1	23301	Agrandamiento en el plano "y".
Nueva coordenada a la izquierda	1	23302	Coordenada "x" de la columna más a la izquierda de la zona en la que será copiada.
Nueva coordenada más baja	1	23303	Coordenada "y" de la fila inferior del área en la que será copiada.

"Llamada" (Call)

RAND USR dirección

Comprobación de errores" (Error Checks)

La rutina vuelve al Basic inmediatamente si cualquiera de las siguientes condiciones son ciertas:

I) Escala horizontal = 0
II) Escala vertical = 0
III) Coordenada más alta mayor que 175.
IV) Nueva coordenada más baja mayor que 175.
V) Coordenada "Y" más baja mayor que la más alta coordenada "Y".
VI) Coordenada "X" izquierda mayor que la coordenada "X" derecha.

Sin embargo, para mantener la rutina en su longitud, no hay comprobación que asegure que la sección copiada caiga en pantalla en buen sitio Simplemente, si lo anterior no se cumple la rutina puede dejar de funcionar. La rutina requiere disponer de gran cantidad de RAM libre, de lo contrario la rutina puede dejar de funcionar.

Comentarios

La rutina no es relocalizable debido a la existencia de un "plot/point" (trazar/punto) que configura una subrutina propia. Esta localizada en la dirección 65033, de lo que se deduce que sólo puede usarse con máquinas de 48 K de RAM. La rutina puede ser reposicionada en memoria, usando el procedimiento dado por la rutina "renumber" (renumerar). Sin embargo, si van a copiarse grandes áreas de la pantalla, se necesita bastante más RAM además de situar la dirección de comienzo tan alta como sea posible.

Listado del código máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld ix, 23296	221 33 0 91
	ld a, 175	62 175
	cp (ix + O)	221 190 0
	ret c	216
	cp (ix + 7)	221 190 7
	ret c	216
	sub a	151
	cp (ix + 4)	221 190 4
	ret z	200
	cp (ix + 5)	221 190 5
	ret z	200
	ld hl, (23296)	42 0 91
	ld b, 1	69
	ld a,l	125
	sub h	148
	ret c	216
	ld (23298),a	50 0 91
	ld e,a	95
	ld hl, (23298)	42 2 91
	ld c,l	77
	ld a,l	125
	sub h	148
	ret c	216
	ld (23298),a	50 2 91
	push bc	197
	ld l,a	111
	ld h, O	38 O
	inc hl	35
	push hl	229

	pop bc	193
	inc e	28
"Add" (suma)	dec e	29
	jr z, remainder	40 3
	add hl,bc	9
	jr add	24 250
"Remainder" (resto)	ld a,l	125
	and 15	230 15
	ld b,a	71
	pop hl	225
	ld c,l	77
	jr nz, save	32 2
"Full" (completar)	ld b,16	6 16
"Save" (guardar)	push hl	229
	call subroutine	205 13* 255*
	and (hl)	166
	jr z, off	40 2
	ld a, 1	62 1
"Off" (desactivar)	pop hl	225
	rra	203 31
	rl e	203 19
	rl d	203 18
	ld a,l	125
	cp (ix + 3)	221 190 3
	jr z, next row	40 6
	dec l	45
"Next Bit" (siguiente bit)	djnz save	16 231
	push de	213
	jr full	24 226
"Next Row" (siguiente fila)	ld l,c	105
	ld a,h	124
	cp (ix + 1)	221 190 1
	jr z, copy	40 3
	dec h	37
	jr next bit	24 241
"Copy" (copiar)	push de	213
	ld b, O	6 O
	ld h,b	96
	ld l,b	104
"Reset" (desactivar)	ld (23306),hl	34 10 91
	ld a,b	120
	or a	183
	jr nz, retrieve	32 3
	pop de	209
	ld b, 16	6 16

"Retrieve" (recuperar)	sub a	151
	dec b	5
	rr d	203 26
	rr e	203 27
	rl a	203 23
	push de	213
	push bc	197
	push af	245
	ld h,l	38 1
"Loop" (lazo)	ld l, l	46 1
"Preserve" (reserva para utilizar después)	ld (23304),hl	34 8 91
	ld a, (23307)	58 11 91
	ld hl, O	33 0 0
	ld de, (23301)	237 91 5 91
	ld d,l	85
"Multiply" (multiplicar)	or a	183
	jr z, calculate	40 6
	add hl,de	25
	dec a	61
	jr multiply	24 249
"Long Jump" (salto relativamente largo)	jr reset	24 208
"Calculate" (calcular)	ld a, (23303)	58 7 91
	add a,l	133
	ld hl, (23304)	42 8 91
	add a,l	133
	dec a	61
	push af	245
	ld a, (23306)	58 10 91
	ld hl, O	33 0 0
	ld de, (23300)	237 91 4 91
	ld d, l	85
"Repeat" (repetir)	or a	183
	jr z, continue	40 4
	add hl,de	25
	dec a	61
	jr repeat	24 249
"Continue" (continuar)	ld a, (23302)	58 6 91
	add a,l	133
	ld hl, (23305)	42 9 91
	add a, l	133
	dec a	61
	ld l,a	111
	pop af	241
	ld h,a	103

	pop af	241
	push af	245
	or a	183
	jr nz, plot	32 7
	call subroutine	205 13* 255*
	cpl	47
	and (hl)	166
	jr Poke	24 4
"Plot" (trazar)	call subroutine	205 13* 255*
	or (hl)	182
"Poke" (introducir valores en direcciones)	ld (hl),a	119
	ld hl, (23304)	42 8 91
	inc l	44
	ld a, (23301)	58 5 91
	inc a	60
	cp l	189
	jr nz, preserve	32 165
	inc h	36
	ld a, (23300)	58 4 91
	inc a	60
	cp h	188
	jr nz, loop	32 155
	pop af	241
	pop bc	193
	pop de	209
	ld hl, (23306)	42 10 91
	inc l	44
	ld a, (23298)	58 2 91
	inc a	60
	cp l	189
	jr nz, long jump	32 164
	ld l, O	46 O
	inc h	36
	ld a, (23296)	58 0 91
	inc a	60
	cp h	188
	jr nz, long jump	32 154
	ret	201
"Subroutine" (subrutina)	push bc	197
	push de	213
	ld a, 175	62 175
	sub h	148
	ld h,a	103
	push hl	229
	and 7	230 7
	add a, 64	198 64

```
                        ld c,a          79
                        ld a,h          124
                        rra             203 31
                        rra             203 31
                        rra             203 31
                        and 31          230 31
                        ld b,a          71
                        and 24          230 24
                        ld d,a          87
                        ld a,h          124
                        and 192         230 192
                        ld e,a          95
                        ld h, c         97
                        ld a,l          125
                        rra             203 31
                        rra             203 31
                        rra             203 31
                        and 31          230 31
                        ld l,a          111
                        ld a,e          123
                        add a,b         128
                        sub d           146
                        ld e,a          95
                        ld d, O         22 O
                        push hl         229
                        push de         213
                        pop hl          225,
                        add hl,hl       41
                        add hl,hl       41
                        add hl,hl       41
                        add hl,hl       41
                        add hl,hl       41
                        pop de          209
                        add hl,de       25
                        pop de          209
                        ld a,e          123
                        and 7           230 7
                        ld b,a          71
                        ld a,8          62 8
                        sub b           144
                        ld b,a          71
                        ld a, 1         62 1
"Rotate" (rotar)        add a,a         135
                        djnz rotate     16 253
                        rra             203 31
                        pop de          209
```

```
pop bc        193
ret           201
```

Funcionamiento

El registro índice *ix* se carga con la dirección de la memoria intermedia de impresora, para usarse como puntero que nos indique las variables. Si la coordenada "Y" más alta o la nueva coordenada "Y" más baja presentan un valor mayor de 175, la rutina vuelve al Basic. Si la escala horizontal o vertical es cero, se vuelve también al Basic.

El registro *h* se carga con la coordenada "Y" más baja, y el registro *l* se carga con la coordenada "Y" más alta. El registro *l* se copia en el registro *b* y en el acumulador, la rutina vuelve al Basic si el resultado es negativo.

El valor del acumulador es cargado en la dirección 23298, para usarlo como un contador.

El registro dual *bc* es guardado después en el área de conservación de registros.

El registro *hl* se carga con el valor en el acumulador, luego incrementado, y copiado después en el registro dual *bc.* Este registro *bc* es sumado a *hl,* un número de veces indicado por el registro *e*, el valor resultante en *hl* es el número de elementos de imagen que serán copiados sobre la pantalla. El acumulador se carga con el valor del registro *l,* y los cuatro bits más a la izquierda se emplazarán a cero. El resultado se copia en el registro *b,* para usarse como contador.

El registro dual *hl* se recoge del área de conservación de registros, luego el registro *l* se copia en el registro *c.* Si el registro *b* contiene un cero, éste se carga con el valor 16, siendo éste el número de bits que integran un registro dual. La "subrutina" es llamada después y el acumulador se carga con el valor "point (l,h)". El registro *de* se hace rotar a la izquierda y el valor del acumulador se carga en el bit más a la derecha del registro *e.*

Si el registro *l* es igual a la coordenada "X" izquierda, la rutina salta a "Next Row". De lo contrario el registro *l* se decrementa, seguido por el registro *b.* Si el registro *b* no contiene un cero, la rutina vuelve al lazo "Save" para depositar el siguiente bit en el registro dual *de.* Si el registro *b* contiene un cero, el registro dual *de* es empujado al área de conservación de registros y se ejecuta un salto a "Full".

En "Next Row" el registro *l* se carga con la coordenada "X" derecha, y el acumulador se carga con el valor que hay en el registro *h.* Si el valor del acumulador es igual a la coordenada "Y" más baja, se ejecuta un salto a "Copy", debido a que el último elemento de imagen a copiar ha sido llevado al registro *de.* En caso contrario el registro *h* se decrementa para apuntar a la siguiente fila y la rutina vuelve a lazo "Next Bit".

En "Copy", *de* es empujado al área de conservación de registros, y los registros *b, h* y *l* son emplazados a cero para usarlos como contadores. El registro dual es POKEado en las direcciónes 23306/7, de forma que *hl* pueda usarse como contador para otros lazos sin que medie el uso del área de conservación de registros. Si el registro *b* contiene un cero, el registro *de* es recogido

del área de conservación de registros y el registro *b* es emplazado a 16, indicando así el número de elementos de imagen almacenados en *de*. El registro *b* se decrementa para indicar que un bit de información va a ser removido desde el registro dual *de*. El bit situado más a la derecha del registro *e* se carga en el acumulador, y el registro dual *e* se rota a la derecha. Los registros duales *de, bc* y *af* son todos empujados al área de conservación de registros, mientras se ejecutan ciertos cálculos.

Los registros *h* y *l* se cargan ambos con el valor 1, para usarlos como contadores, y el registro dual es introducido en las direcciones 23304/5. El acumulador se carga con el valor del octeto de la dirección 23307, siendo éste uno de los contadores guardados anteriormente. El registro dual *de* se carga con la escala vertical. Éste luego se multiplica por el valor que hay en el acumulador, el resultado es introducido en *hl*. Éste es sumado junto con la nueva coordenada "Y" más baja en el registro acumulador. El octeto que hay en la dirección 23304 es sumado luego al acumulador, y el resultado es decrementado.

El acumulador contiene ahora la coordenada "Y" del siguiente elemento de imagen que será trazado. Esto es guardado en el área de conservación de registros; asimismo, la coordenada "X" se calcula por un proceso muy similar. Cuando la coordenada "X" es calculada, se carga en el registro *l*. La coordenada "Y" se recupera desde el área de conservación de registros y cargada en el registro *h*. El acumulador se emplaza al último valor que tuviere en el área de conservación de registros. Si este valor es 1, el punto (x,y) deberá ser trazado; de lo contrario no debe trazarse. Se hace una llamada a la subrutina y la acción consiguiente se ejecuta.

El registro dual *hl* se carga a los contadores de lazo, almacenados en las direcciones 23304/5. El registro *l* se incrementa, y si éste no contiene el valor (1 + escala vertical), la rutina retrocede al lazo "Preserve". Se incrementa el registro *h*, y si éste no contiene el valor (1 + escala horizontal) se ejecuta un salto a "Loop".

Los registros duales *af, bc* y *de* son recogidos del área de conservación de registros y el registro dual *hl* es cargado con el segundo conjunto de contadores de lazo, el cual está almacenado en las direcciones 23306/7. El registro *l* se incrementa, a continuación se ejecuta un salto a "Reset" si el resultado no es igual a (coordenada "X" derecha — coordenada "X" izquierda + 1). El registro *l* se emplaza a cero, siendo éste el valor original del contador de lazo. El registro *h* se incrementa, y la rutina vuelve a "Reset" si el resultado no es igual a (coordenada "Y" mas alta — coordenada "Y" más baja + 1). La rutina vuelve al Basic.

La "subrutina" es idéntica a la que se usa en la rutina "rellenando zonas".

7. RUTINAS DE MANIPULACION DE PROGRAMAS

Borrar un Bloque de Programa

Longitud: 42
Número de variables: 2
Prueba de la suma: 5977

Operación

Esta rutina borra bloques del programa Basic entre dos líneas especificadas por el usuario.

Variables

Nombre	*Longitud*	*Posición*	*Comentario*
N.° de línea de comienzo	2	23296	Primera línea a borrar.
N.° de línea de final.	2	23298	Ultima línea a borrar.

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Si cualquiera de los errores siguientes ocurren la rutina se para sin borrar ninguno de los bloques del programa Basic:

I) Si el número dado a la última línea a borrar es menor que el dado a la primera línea.
II) Cuando no existe un programa en Basic entre las dos líneas dadas.
III) Si alguna de las líneas especificados son cero.

Comentarios

Esta rutina es bastante lenta para borrar un bloque grande de líneas del programa, pero es más rápido usarla que ir borrándolas a mano. No asignar valores de líneas mayor que 9999.

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, (23296)	42 0 91
	ld de, (23298)	237 91 2 91
	ld a,h	124
	or l	181
	ret z	200
	ld a,d	122

	or e	179
	ret z	200
	push de	213
	call 6510	205 110 25
	ex (sp), hl	227
	inc hl	35
	call 6510	205 110 25
	pop de	209
	and a	167
	sbc hl,de	237 82
	ret z	200
	ret c	216
	ex de,hl	235
"Next Chr:" (siguiente carácter)	ld a,d	122
	or e	179
	ret z	200
	push de	213
	push hl	229
	call 4120	205 24 16
	pop hl	225
	pop de	209
	dec de	27
	jr next chr	24 243

Funcionamiento

Los registros duales *hl* y *de* se cargan con los números de línea principio y final respectivamente. Los valores son comprobados y si alguno o ambos son cero, la rutina vuelve al Basic

La rutina de la ROM que está en la dirección 6510 es llamada a continuación, y devuelve la dirección de la primera línea. Será llamada de nuevo con el fin de encontrar la dirección del carácter de después del comando "Enter" en la línea final. El registro dual *hl* es emplazado con el valor de la diferencia entre las dos direcciones (primera línea y última en borrar), si esta diferencia resulta ser cero o negativa, la rutina vuelve al Basic.

El contenido del registro dual *hl* se transfiere al registro *de,* para usarse como contador. Si el contador es cero, la rutina termina, caso de que no lo fuere, se llama a la rutina que comienza en la dirección 4120 (en la ROM), lo cual hace que se borre un carácter. La rutina retrocede a "Next Char:".

Intercambio de Claves

Longitud: 46
Número de variables: 2
Prueba de la suma: 5000

ALTERNATIVA DE LA RUTINA SPRITES

```
        ORG   61000
        LD    HL,30000
        LD    (59000),HL
        LD    A,32
        LD    (59002),A
        LD    HL,1
        LD    (59003),HL
        LD    HL,16384
        LD    DE,16385
        LD    BC,2047
        LD    (HL),L
        LDIR
        LD    HL,30000
        LD    DE,16384
        LD    A,(59002
        LD    B,A
BUCLE7  PUSH  BC
        PUSH  DE
        LD    B,8
BUCLE6  PUSH  BC
        PUSH  DE
        LD    B,8
BUCLE5  PUSH  BC
        PUSH  DE
        LD    BC,(59003)
        NOP
        LDIR
        NOP
        POP   DE
        INC   D
        POP   BC
        DJNZ  BUCLE5
        LD    BC,32
        POP   DE
        EX    DE,HL
        ADD   HL,BC
        EX    DE,HL
        POP   BC
        DJNZ  BUCLE6
        LD    BC,(59003)
        POP   DE
        EX    DE,HL
        ADD   HL,BC
        EX    DE,HL
        POP   BC
        DJNZ  BUCLE7
        RET
```

Operación

Cada vez que aparece, en un programa Basic, un carácter cambiará éste por otro carácter especificado. Por ejemplo, todas las sentencias "PRINT" pueden cambiarse por "LPRINT".

Variables

Nombre	*Longitud*	*Posición*	*Comentario*
Antiguo carácter	1	23296	Carácter a reemplazar
Nuevo carácter	1	23297	Carácter a introducir.

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Si no hay un programa Basic en memoria o alguno de los caracteres especificados tienen un código menor que 32, la rutina vuelve al Basic.

Comentarios

Esta rutina es muy rápida, pero cuanto más largo sea el programa en Basic, más tiempo empleará en su ejecución.

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld, bc, (23296)	237 75 0 91
	ld a,31	62 31
	cp b	184
	ret nc	208
	cp c	185
	ret nc	208
	ld hl, (23635)	42 83 92
"Next Chr:" (siguiente carácter)	inc hl	35
	inc hl	35
	inc hl	35
"Check:" (comprobar)	ld de, (23627)	237 91 75 92
	and a	167
	sbc hl,de	237 82
	ret nc	208
	add hl,de	25
	inc hl	35
	ld a, (hl)	126
	inc hl	35
	cp 13	254 13
	jr z, next chr	40 237
	cp 14	254 14

	jr nz, compare	32 3
	inc hl	35
	jr next chr	24 230
"Compare" (comparar)	dec hl	43
	cp c	185
	jr nz, check	32 229
	ld (hl), b	112
	jr check	24 226

Funcionamiento

Los registros *b* y *c* se cargan con los caracteres nuevos y antiguos respectivamente. Si algún carácter tiene un còdigo menor que 32, la rutina vuelve al Basic.

El registro dual *hl* se carga con la dirección de comienzo del programa Basic. Luego se incrementa el registro dual *hl* y lo compara con la dirección del área de variables. Si el registro *hl* no es menor que la dirección de las variables, la rutina vuelve al Basic.

El registro dual *hl* se incrementa para apuntar al siguiente carácter. El código de este carácter se carga en el acumulador y el registro *hl* se incrementa de nuevo. Si el valor del acumulador es 13 ó 14 (es decir, "Enter" o "Number") la rutina salta atrás hasta "Next Chr:" y el registro *hl* se incrementa para apuntar al siguiente carácter. Si el acumulador no contiene 13 ó 14, el valor almacenado es comparado con el "antiguo carácter". Si se establece la igualdad, este carácter se reemplaza por el "nuevo carácter".

La rutina salta luego para comprobar si se ha llegado al final del programa.

Elimina las Instrucciones "REM" (Rem Kill)

Longitud: 132
Número de variables: 0
Prueba de la suma: 13809

Operación

Esta rutina elimina todas las sentencias "REM" que haya en un programa Basic residente en memoria.

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Si no existe en memoria un programa en Basic, la rutina deberá devolver el control sin tener ningún efecto.

Comentarios

La rutina ROM usada para borrar caracteres no es ciertamente muy rápida y, por lo tanto, emplea algún tiempo en su ejecución.

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, (23635)	42 83 92
	jr check	24 31
"Next Line" (siguiente línea)	push hl	229
	inc hl	35
	inc hl	35
	ld c, (hl)	78
	inc hl	35
	ld b, (hl)	70
"Next Chr" (siguiente carácter)	inc hl	35
	ld a,(hl)	126
	cp 33	254 33
	jr c, next chr	56 250
	cp 234	254 234
	jr nz, search	32 26
	inc bc	3
	inc bc	3
	inc bc	3
	inc bc	3
	pop hl	225
"Delete Line" (borrar línea)	push bc	197
	call 4120	205 24 16
	pop bc	193
	dec bc	11
	ld a,b	120
	or c	177
	jr nz, delete line	32 246
"Check" (comprobar)	ld de, (23627)	237 91 75 92
	and a	167
	sbc hl,de	237 82
	ret nc	208
	add hl,de	25
	jr next line	24 214
"Search" (buscar)	inc hl	35
	ld a, (hl)	126
	cp 13	254 13
	jr nz, not enter	32 8
"Enter Found" introducir —encontrado—)	pop hl	225
	add hl,bc	9
	inc hl	35

	inc hl	35
	inc hl	35
	inc hl	35
	jr check	24 231
"Not Enter" (no introducir)	cp 14	254 14
	jr nz, not number	32 7
	inc hl	35
	inc hl	35
	inc hl	35
	inc hl	35
	inc hl	35
	jr search	24 231
"Not Number" (no número)	cp 33	254 33
	jr c, search	56 227
	cp 34	254 34
	jr nz, not quote	32 8
"Find Quote" (encontradas comillas)	inc hl	35
	ld a, (hl)	126
	cp 34	254 34
	jr nz, find quote	32 250
	jr search	24 215
"Not quote" (no hay comillas)	cp 58	254 58
	jr nz, search	32 211
	ld d,h	84
	ld e, l	93
"Find Enter" (encontrar el "Enter")	inc hl	35
	ld a, (hl)	126
	cp 13	254 13
	jr z, enter found	40 209
	cp 33	254 33
	jr c, find enter	56 246
	cp 234	254 234
	jr nz, not quote	32 236
	ld h,d	98
	ld l,e	107
"Delete Chr" (borrar carácter)	push bc	197
	call 4120	205 24 16
	pop bc	193
	dec bc	11
	ld a, (hl)	126
	cp 13	254 13
	jr nz, delete chr	32 245
	pop hl	225
	inc hl	35
	inc hl	35

```
ld (hl),c        113
inc hl           35
ld (hl),b        112
dec hl           43
dec hl           43
dec hl           43
jr check         24 160
```

Funcionamiento

El registro dual *hl* se carga con la dirección de comienzo del área de programa en Basic, se ejecuta luego un salto a la rutina que comprueba si es ya el final del área de programa. Si lo es, se vuelve a BASIC.

La rutina salta a "Next Line". Esta sección guarda la dirección que hay en *hl* en el área de conservación de registros para usarla más tarde, luego carga el registro *bc* con la longitud de la línea Basic que ha sido encontrada. La rutina "Next Chr" incrementa la dirección en *hl* y carga el acumulador con el carácter almacenado en esa dirección. Si este carácter tiene un código menor que 33, indicando que hay un espacio o carácter de control, la rutina salta atrás para repetir de nuevo esta sección. Si el carácter encontrado no es la expresión "REM" se ejecuta un salto a "Search".

Si encuentra un sentencia "REM" el registro *bc* se incrementa cuatro veces, de forma que pueda usarse como un contador, el registro dual *hl* es removido de la parte alta del área de conservación de registros. A continuación, los *bc* caracteres a partir de la dirección *hl* se eliminarán, usando la rutina de ROM que comienza en la dirección 4120. La rutina luego "baja" recorriendo la rutina de comprobación una vez más.

Si se ejecuta un salto a la rutina "Search" el registro *hl* se incrementa para apuntar al siguiente caráceter y éste se carga en el acumulador. Si éste resulta ser el carácter "Enter" el registro *hl* se recarga, desde el área de conservación de registros, incrementándole seguidamente para que apunte al comienzo de la línea siguiente, luego se ejecuta un salto a "Check".

Si el acumulador contiene el carácter "Number" (código 14) el registro *hl* se incrementa para apuntar al primer carácter después de ser almacenado el número y el proceso de búsqueda se repite.

Posteriormente se ejecuta una comprobación para aquellos caracteres cuyos códigos son menores que 33. Caso de encontrar uno se ejecutará un salto atrás a "Search".

Si se encuentra un carácter "comillas" (34), la rutina vuelve al lazo hasta que una segunda "comilla" se encuentre, y así la búsqueda continúa. Si el carácter que fue encontrado no es (:) (dos puntos), lo cual indicaría una línea de programa multi-sentencia, la búsqueda se repite de nuevo. El registro *hl* se transfiere al registro *de* para guardar la dirección del carácter (:) (dos puntos), luego se incrementa el registro *hl* para apuntar al siguiente carácter. Si este carácter es un "Enter", se ejecuta un salto a "Enter-Found" (Enter-encontrado); por lo contrario, en el caso de un carácter de control o espacio, la rutina volvería a "Find Enter" (Busear Enter).

Si el carácter no es una expresión "REM" se ejecuta un salto a "Not Quote". Si se encuentra una expresión "REM", el registro *hl* se carga con la dirección de los últimos (:) (dos puntos) encontrados, luego todos los caracteres tomados de *hl* hasta la siguiente expresión "Enter" son eliminados. Una vez hecho esto, los punteros se corrigen para apuntar a las líneas, el registro *hl* se emplaza al comienzo de la línea y se ejecuta un salto a "Check".

Creación de Sentencias "REM"

Longitud: 85

Número de variables: 3

Prueba de la suma: 9526

Operación

Esta rutina crea sentencias "REM", en una línea específica, conteniendo un número de caracteres. El caracter lo elige el usuario.

Variables

Nombre	*Longitud*	*Posición*	*Comentario*
Número de línea	2	23296	Línea en la que se insertará la "REM".
Numero de caracteres	2	23298	Número de caracteres después de "REM".
Código de los caracteres	1	23300	Código de los caracteres que siguen a "REM".

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Si el número de línea dado es cero, o más que 9999, o también en el caso en que el mismo número exista ya, la rutina vuelve al Basic.

Comentarios

Esta rutina no comprueba el hecho de que haya suficiente memoria libre para insertar una línea nueva. Sin embargo, esto podría hacerse antes de poner en funcionamiento la rutina, mediante la rutina "Memory Left" que que hay en este libro.

Los caracteres a introducir después de la sentencia "REM" deberán tener, preferiblemente, códigos mayores que 31 ya que los caracteres de control (0 - 31) podrían entrar en conflicto con la rutina "List" que hay en la ROM.

La rutina ROM, que es llamada para insertar caracteres, es precisamente bastante lenta, de manera que esta rutina tomará un tiempo relativamente largo.

La sentencia "REM" creada por el uso de esta rutina puede usarse para almacenar código máquina o datos que han de cargarse en ese lugar.

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, (23296)	42 0 91
	ld, a,h	124
	or l	181
	ret z	200
	ld de, 10000	17 16 39
	and a	167
	sbc hl,de	237 82
	ret nc	208
	add hl,de	25
	push hl	229
	call 6510	205 110 25
	jr nz, create	32 2
	pop hl	225
	ret	201
"Create" (crear)	ld bc, (23298)	237 75 2 91
	push bc	197
	push bc	197
	ld a, 13	62 13
	call 3976	205 136 15
	inc hl	35
	pop bc	193
"Next Chr" (siguiente carácter)	push bc	197
	ld a,b	120
	or c	177
	jr z, insert REM	40 11
	ld a, (23300)	58 4 91
	call 3976	205 136 15
	inc hl	35
	pop bc	193
	dec bc	11
	jr next chr	24 240
"Insert REM" (insertar REM)	pop bc	193
	ld a, 234	62 234
	call 3976	205 136 15
	inc hl	35
	pop bc	193
	inc bc	3
	inc bc	3
	ld a,b	120
	push bc	197

call 3976	205 136 15
pop bc	193
inc hl	35
ld a,c	121
call 3976	205 136 15
inc hl	35
pop bc	193
ld a,c	121
push bc	197
call 3976	205 136 15
pop bc	193
inc hl	35
ld a,b	120
jp 3976	195 136 15

Funcionamiento

El registro dual *hl* se carga con el número de línea especificado. Se compara con el valor cero, y si se encuentra una igualdad, la rutina vuelve al Basic. También, si el registro *hl* contiene un número mayor que 9999 (que corresponde al valor más alto que puede tomar un número de línea), se ejecuta una vuelta al Basic.

Se llama a la rutina ROM que devuelve en el registro *hl* la dirección de la línea cuyo número estaba previamente en el registro *hl*. Si el indicador cero está activado, es que existe allí una línea, de manera que la rutina vuelve al Basic.

Si el indicador cero no esta activo se ejecuta un salto a "Create". El registro *bc* se carga con el número de caracteres que serán insertados despues de "REM", y este número se guarda en el área de conservación de registros. El acumulador se carga con un 13, que es el código del carácter "Enter". La rutina ROM en la dirección 3976 es llamada con el fin de insertar el carácter "Enter". El registro *bc* se recupera desde el área de conservación de registros. Después volvemos a guardar el registro *bc* en el área de conservación de registros, este registro *bc* es puesto a prueba para ver si hay que insertar algún carácter más. Caso de que no, se ejecuta un salto a "Insert REM". Si otro carácter ha de ser insertado, el acumulador se carga con el código especificado y la rutina ROM ubicada en 3976 se usa para insertarlo. El contador (bc) se decrementa y la rutina vuelve a ejecutar el lazo para comprobar si el registro *bc* es cero. Una vez que la rutina alcanza "Insert REM", una expresión "REM" se inserta por medio de la misma rutina ROM. El registro *bc* se carga con la longitud de esa nueva línea, y se crean los punteros para esa línea. El número de línea se remueve desde el área de conservación de registros, y se inserta, finalmente antes de que vuelva al Basic.

Compactación de Programas

Longitud: 71

Número de variables: 0

Prueba de la suma: 7158

Operación

Esta rutina elimina todos los espacios y caracteres de control no necesarios que puedan existir en un programa Basic, incrementando de esta forma la cantidad de memoria RAM disponible.

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Si no hay en memoria un programa en Basic, la rutina vuelve inmediatamente al Basic.

Comentarios

Esta rutina asume que todas las sentencias "REM" han sido ya previamente removidas del programa Basic. Sin embargo, si esto no es así, el computador no se queda colgado. El tiempo que la rutina toma en terminar es proporcional a la longitud del programa Basic que hay en memoria.

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, (23635)	42 83 92
"Next Line" (siguiente línea)	inc hl	35
	inc hl	35
"Check" (comprobar)	ld de, (23627)	237 91 75 92
	and a	167
	sbc hl,de	237 82
	ret nc	208
	add hl,de	25
"Length" (longitud)	push hl	229
	ld c, (hl)	78
	inc hl	35
	ld b, (hl)	70
"Next Byte" (siguiente octeto)	inc hl	35
"Load" (cargar)	ld a, (hl)	126
	cp 13	254 13
	jr nz, number	32 8
"Restore" (volver a almacenar)	pop hl	225
	ld (hl),c	113
	inc hl	35
	ld (hl),b	112
	add hl,bc	9
	inc hl	35
	jr next line	24 227
"Number" (número)	cp 14	254 14

	jr nz, quote	32 7
	inc hl	35
	inc hl	35
	inc hl	35
	inc hl	35
	inc hl	35
	jr next byte	24 231
"Quote" (comillas)	cp 34	254 34
	jr nz, control	32 12
"Find Quote" (encontrar comillas)	inc hl	35
	ld a, (hl)	126
	cp 34	254 34
	jr z, next byte	40 221
	cp 13	254 13
	jr z, restore	40 223
	jr find quote	24 244
"Control" (control)	cp 33	254 33
	jr nc, next byte	48 211
	push bc	197
	call 4120	205 24 16
	pop bc	193
	dec bc	11
	jr load	24 204

Funcionamiento

El registro dual *hl* se carga con la dirección del programa Basic. Si el registro *hl* se incrementa luego dos veces, de forma que apunte a los dos octetos que almacenan la longitud de la línea siguiente. El registro dual *de* se carga con la dirección del área de variables. Si el registro *hl* no es menor que el *de,* la rutina vuelve al Basic debido a que se ha alcanzado el fin del programa.

La dirección en *hl* se guarda en el área de conservación de registros, el registro *bc* se carga con la longitud de la línea presente, y el registro *hl* se incrementa para apuntar al octeto siguiente que hay en la línea. El octeto en *hl* se carga en el acumulador. Si este no contiene un trece, se ejecutará un salto a "Number".

Para alcanzar "Restore, se habrá tenido que encontrar el final de la presente línea. La dirección de los punteros de la línea se carga desde el área de conservación de registros en el registro *hl*, se inserta la longitud presente. La longitud de la línea se suma a *hl*, *hl* se incrementa y la rutina vuelve a "Next Line".

Si la rutina alcanza "Number", el acumulador es comprobado de tal manera que se pueda ver si éste contiene el carácter "Number" (14). Si esto ocurre, *hl* se incrementa cinco veces, de forma que el número siguiente no se cambie, y se pueda ejecutar el salto a "Next Byte".

Si el acumulador no contiene el código del carácter "comillas" la rutina salta a "Control". Si se encuentra un carácter "comillas" la rutina salta hasta

que se alcance el final de línea, o hasta llegar a otro carácter "comillas". En el primer caso se ejecuta un salto a "Restore", y en el segundo caso el salto se hará a "Next Byte" (siguiente octeto).

En "Control" se comprueba el carácter para ver si éste tiene un código menor que 33. Caso de que no, la rutina vuelve al lazo "Next Byte".

Si un espacio o carácter de control es encontrado, la rutina ROM ubicada en la dirección 4120 es reclamada para que actúe sobre ese carácter, eliminándolo. La longitud de línea, que está mantenida en *bc,* se decrementa. Luego un salto es ejecutado con destino en "Load".

Cargar Código Máquina en Sentencias "DATA"

Longitud: 179
Número de variables: 2
Prueba de la suma: 19181

Operación

Esta rutina produce una sentencia "DATA" en la línea uno del programa Basic y luego la rellena con los datos extraídos de memoria.

Variables

Nombre	*Longitud*	*Posición*	*Comentario*
Comienzo de datos	2	23296	Dirección desde la que se ha de copiar.
Longitud de dato	2	23298	Número de octetos a copiar.

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Si el número de octetos a copiar es cero o en su defecto existe ya la línea uno, el resultado vuelve al Basic inmediatamente. La rutina no comprueba si hay espacio disponible en la memoria "RAM" para una nueva línea, de manera que ello se ha de hacer manualmente.

La rutina requiere diez octetos por cada octeto de datos, más cinco para números de línea, punteros, etc. Sin embargo, la rutina ROM usada también utiliza a su vez un gran área de trabajo, así que este es un punto a tener en cuenta. Si no hay suficiente memoria disponible, los punteros de línea podrían no estar correctamente emplazados y el listado Basic, por consecuencia, resultaría erróneo.

Comentarios

El tiempo requerido por esta rutina es proporcional a la longitud de memoria que será copiada.

Listado del Código Máquina

Etiqueta	*Lenguaje Ensamblador*	*Números a introducir*
	ld de, (23296)	237 91 0 91
	ld bc, (23298)	237 75 2 91
	ld a,b	120
	or c	177
	ret z	200
	ld hl, (23635)	42 83 92
	ld a, (hl)	126
	cp O	254 0
	jr nz, continue	32 6
	inc hl	35
	ld a, (hl)	126
	cp 1	254 1
	ret z	200
	dec hl	43
"Continue" (continuar)	push hl	229
	push bc	197
	push de	213
	sub a	151
	call 3976	205 136 15
	ex de,hl	235
	ld a, 1	62 1
	call 3976	205 136 15
	ex de,hl	235
	call 3976	205 136 15
	ex de,hl	235
	call 3976	205 136 15
	ex de,hl	235
	ld a, 228	62 228
	call 3976	205 136 15
	ex de,hl	235
"Next Byte" (siguiente octeto)	pop de	209
	ld a, (de)	26
	push de	213
	ld c,47	14 47
"Hundreds" (centenas)	inc c	12
	ld b, 100	6 100
	sub b	144
	jr nc, hundreds	48 250
	add a,b	128
	ld b,a	71
	ld a,c	121
	push bc	197
	call 3976	205 136 15

```
                                ex de,hl         235
                                pop bc           193
                                ld a,b           120
                                ld c,47          14 47
"Tens" (decenas)                inc c            12
                                ld b,10          6 10
                                sub b            144
                                jr nc, tens      48 250
                                add a,b          128
                                ld b,a           71
                                ld a,c           121
                                push bc          197
                                call 3976        205 136 15
                                pop bc           193
                                ex de,hl         235
                                ld a,b           120
                                add a,48         198 48
                                call 3976        205 136 15
                                ex de,hl         235
                                ld a,14          62 14
                                ld b, 6          6 6
"Next Zero" (siguiente cero)    push bc          197
                                call 3976        205 136 15
                                pop bc           193
                                ex de,hl         235
                                sub a            151
                                djnz next zero   16 247
                                pop de           209
                                push hl          229
                                dec hl           43
                                dec hl           43
                                dec hl           43
                                ld a, (de)       26
                                ld (hl),a        119
                                pop hl           225
                                inc de           19
                                pop bc           193
                                dec bc           11
                                ld a,b           120
                                or c             177
                                jr z, enter      40 10
                                push bc          197
                                push de          213
                                ld a, 44         62 44
                                call 3976        205 136 15
                                ex de, hl        235
```

	jr next byte	24 173
"Enter" (introducir)	ld a,13	62 13
	call 3976	205 136 15
	pop hl	225
	ld bc, 0	1 0 0
	inc hl	35
	inc hl	35
	ld d,h	84
	ld e,l	93
	inc hl	35
"Pointers" (punteros)	inc hl	35
	inc bc	3
	ld a, (hl)	126
	cp 14	254 14
	jr nz, end?	32 12
	inc bc	3
	inc bc	3
	inc bc	3
	inc bc	3
	inc bc	3
	inc hl	35
	inc hl	35
	inc hl	35
	inc hl	35
	inc hl	35
	jr pointers	24 237
"End?" (es el final?)	cp 13	254 13
	jr nz pointers	32 233
	ld a,c	121
	ld (de),a	18
	inc de	19
	ld a,b	120
	ld (de),a	18
	ret	201

Funcionamiento

El registro dual *de* se carga con la dirección de los octetos a copiar y el registro dual *bc* se carga con el número de octetos que serán copiados. Si el registro *bc* contiene un cero, la rutina vuelve al Basic inmediatamente.

El registro dual *hl* se carga con la dirección del programa Basic.

El acumulador se carga con el octeto almacenado en la dirección contenida en *hl*. Este es el octeto superior o alto del número de línea. Si éste no contiene cero, es que la línea uno no existe y por consiguiente la rutina salta al "continue". Si el octeto superior contiene un cero, el acumulador se carga con el octeto inferior o bajo. Si éste está emplazado a uno, la línea uno ya existe y por lo tanto la rutina vuelve al Basic.

La dirección del octeto superior del número de línea se guarda en el área de conservación de registros. También se guarda el número de octetos que serán copiados, seguido de la dirección de los datos.

El acumulador se carga con cero —que es el octeto superior del número de la nueva línea. Llamando a la rutina en ROM que está ubicada a partir de la dirección 3976, podremos insertar el carácter contenido en el acumulador en la dirección expresada por *hl*. El registro *hl* se emplaza al valor contenido antes de esta operación. El acumulador se carga con uno, y éste es insertado tres veces. La primera es el octeto inferior del número de línea; las otras dos vienen a ser el puntero de línea. El acumulador se carga luego con el código de la expresión "DATA" y a continuación se inserta ésta.

La dirección del siguiente octeto de datos se recoge del área de conservación de registros y cargado en *de*. El acumulador se carga con este octeto, y el registro *de* es llevado de nuevo al área de conservación de registros.

El registro *c* se carga con el código del carácter "O" menos uno. El registro *c* se incrementa y el registro *b* se carga con 100. El registro *b* se sustrae del acumulador y si el resultado no es negativo la rutina vuelve al lazo "Hundreds".

El registro *b* es sumado una vez al acumulador de forma que éste contenga un valor positivo. Este valor es cargado luego en el registro *b*. El acumulador se carga con el contenido de *c*, y el registro dual *bc* se guarda en el área de conservación de registros. La rutina en la dirección 3976 (ROM) inserta el carácter contenido en el acumulador, en la dirección expresada en *hl*. El registro dual *bc* se recoge del área de conservación de registros y el acumulador se carga con el valor del registro *b*. El proceso anterior se repite para $b=10$. Luego, se incrementa el acumulador por 48 y se inserta el carácter resultante.

La rutina arriba indicada ha insertado el valor decimal del octeto de datos encontrado, en la sentencia "DATA", ahora deberá insertarse su representación binaria. Estos es señalado por la expresión "Number", código 14, siendo éste el primero en entrar seguido de cinco ceros. El valor del octeto que está siendo copiado es cargado para reemplazar el tercer cero. El registro *de* es incrementado para apuntar al siguiente octeto de datos. El número de octetos a copiar se transfiere desde el área de conservación de registros al registro *bc*, y este se decrementa a continuación. Si el resultado es cero, se ejecuta un salto a "Enter"; de lo contrario, los registros *bc* y *de* son devueltos al área de conservación de registros insertándose una coma en la sentencia DATA, y la rutina ejecuta el lazo "Next Byte".

En "Enter", una expresión del tipo "Enter" es insertada con el fin de señalar el final de la sentencia "DATA". El registro *hl* se carga con la dirección del comienzo de la línea, el registro *bc* se emplaza a cero. El registro dual se incrementa para apuntar al octeto inferior del puntero de línea, y esta nueva dirección se copia en *de*. El registro *hl* se incrementa para apuntar al octeto superior del puntero de línea. El registro *hl* y *bc* se incrementa, y el acumulador se carga con el carácter en la dirección almacenada en *hl*.

Si el acumulador contiene 14, es que se ha encontrado un número, y por consiguiente ambos registros, *hl* y *bc*, serán incrementados cinco veces para

apuntar al primer carácter de después del número, la rutina ejecuta luego el lazo "Pointers".

Si el acumulador no contiene 14, pero si contiene 13, retrocede de salto hacia "pointers".

Llegados a este punto, la expresión "Enter" señalando el final de la línea deberá de encontrarse casi a continuación. El registro *bc* contiene ahora la longitud de la línea, de manera que ésta sea cargada en el puntero de línea, cuya dirección se encuentra en *de*.

La rutina vuelve al Basic.

Convierte Minúsculas en Mayúsculas

Longitud: 41

Número de variables: 0

Prueba de la suma: 4683

Operación

Esta rutina convierte todos los caracteres en minúsculas, que haya en un programa en Basic, por caracteres en mayúsculas. El caso contrario también lo realiza.

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Si no hay en memoria un programa en Basic, la rutina vuelve al programa supervisor Basic.

Comentarios

Para cambiar esta rutina de forma que la transformación sea de mayúsculas a minúsculas, se han de cambiar lo números señalados con asterisco que a continuación se indican:

Cambiar 96* por 64
Cambiar 90** por 122

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, (23635)	42 83 92
	ld de, (23627)	237 91 75 92
"Jump" (salto)	inc hl	35
	inc hl	35
	inc hl	35
	inc hl	35
"Changed" (cambiado a...)	inc hl	35
"Next Byte" (siguiente octeto)	and a	167

sbc hl,de	237 82
ret nc	208
add hl,de	25
ld a, (hl)	126
cp 13	254 13
jr z, jump	40 241
cp 14	254 14
inc hl	35
jr z, jump	40 236
sub 96	214 96*
jr c, next byte	56 237
sub 26	214 26
jr nc, next byte	48 233
add a,90	198 90**
dec hl	43
ld (hl), a	119
jr changed	24 226

Funcionamiento

El registro dual *hl* se carga con la dirección del programa Basic y el registro dual *de* se carga con la dirección del área de variables. El registro *hl* se incrementa para saltar sobre el "número de línea/punteros". Si *hl* no es menor que *de,* la rutina vuelve al Basic, toda vez que se ha llegado al final del programa.

El acumulador se carga con el octeto almacenado en *hl.* Si este octeto es un cańacter "Enter" la rutina vuelce a "Jump". Si el octeto es la expresión "Number", la rutina vuelve también a "Jump", teniendo incrementado ya el registro *hl.* Así es como son evitados los cinco octetos de después del carácter 14.

Al acumulador se le sustrae el 96. Si el resultado es negativo, la rutina salta a "Next Byte" debido a que el carácter no puede ser una minúscula. Del acumulador se sustrae el 26. Si el resultado no es negativo, se ejecuta un salto a "Next Byte", ya que el carácter tiene un código demasiado alto como para ser una minúscula. Se suma 90 al acumulador para dar el código de su homólogo en mayúscula. El registro *hl* se decrementa para apuntar al carácter que será reemplazado. Esta dirección es cargada con el valor que hay en el acumulador y se ejecuta un salto a "Changed".

8. RUTINAS - HERRAMIENTA

Renumerar (Renumber)

Longitud: 382
Número de variables: 2
Prueba de la suma: 41423

Operación

Esta rutina renumera un programa Basic incluyendo cualquier "GOTO", "GOSUB", etc.

Variables

Nombre	*Longitud*	*Posición*	*Comentario*
Número de la primera línea	2	23296	Número de la primera línea al comenzar a funcionar el programa.
Paso	2	23298	Diferencia entre números consecutivos de línea.

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Si el número de la primera línea es cero, o el paso es cero, la rutina vuelve al Basic inmediatamente. Si no hay en la RAM un programa en Basic, la rutina vuelve al Basic. Cualquier número o números de líneas que son resultado de un cálculo (ejemplo: GOTO 7*A); número incluyendo puntos decimales, números menores que cero (ejemplo: GOTO —1) o número mayores que 9999 (por ejemplo: GOTO 20170), todos estos casos serán ignorados. Si el paso es demasiado largo, los números de línea puede estar repetidos y el programa se contamina. La rutina incrementa la longitud del programa en Basic existente en la RAM, por lo que la comprobación de la existencia de algo más de memoria disponible se debe hacer siempre.

Comentarios

El tiempo empleado por esta rutina es proporcional a la longitud del programa Basic que hay en la RAM.

La rutina no es relocalizable y normalmente deberá entrar en la posición de memoria 32218. Esta dirección puede cambiarse siguiendo este procedimiento:

I) Let X = nueva dirección — 32218
II) Let H = INT (x/256)
Let L = x — 256*h
III) Por cada par de números marcados con asterisco que se encuentren en el listado, operar de modo siguiente:

Let LI = L + el primer número
Let HI = H + el segundo número
Si LI es más de 255 Let HI = HI + 1
Let LI = LI — 256
El par de números se deben reemplazar por LI y HI.

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, (23296)	42 0 91
	ld a, h	124
	or L	181
	ret z	200
	ld hl, (23298)	42 2 91
	ld a,h	124
	or L	181
	ret z	200
	ld hl, (23635)	42 83 92
	ld de, (23296)	237 91 0 91
"Next Line" (siguiente línea)	call check	205 76* 127*
	jr nc, find GOTO	48 22
	ld b, (hl)	70
	ld (hl),d	114
	inc hl	35
	ld c, (hl)	78
	ld (hl),e	115
	inc hl	35
	ld (hl),c	113
	inc hl	35
	ld (hl),b	112
	inc hl	35
	push hl	229
	ld hl, (23298)	42 2 91
	add hl,de	25
	ex de, hl	235
	pop hl	225
	call end of line	205 65* 127*
	jr next line	24 229
"Find GOTO" (encontrar la sentencia GOTO)	ld hl, (23635)	42 83 92
	inc hl	35
	inc hl	35
	inc hl	35
	inc hl	35
"Search" (buscar)	call find	205 235* 126*
	jp nc, restore	210 184* 126*
	ld d,h	84
	ld e, l	93

	ld b, O	6 0
"Next Digit" (siguiente dígito)	inc b	4
	inc hl	35
	ld a, (hl)	126
	cp 46	254 46
	jr nz, continue	32 3
"Find Next" (encontrar el siguiente)	ex de,hl	235
	jr search	24 236
"Continue" (continuar)	cp 14	254 14
	jr nz, next digit	32 242
	inc hl	35
	inc hl	35
	inc hl	35
	inc hl	35
	inc hl	35
	inc hl	35
	ld a, (hl)	126
	cp 58	254 58
	jr z, found	40 4
	cp 13	254 13
	jr nz, find next	32 234
"Found" (encontrado)	ld a, b	120
"Compare" (comparar)	cp 4	254 4
	jr z, calculate	40 16
	jr nc, find next	48 227
	push de	213
	ld h,d	98
	ld l,e	107
	push af	245
	ld a, 48	62 48
	call 3976	205 136 15
	pop af	241
	inc a	60
	pop de	209
	jr compare	24 236
"Calculate" (calcular)	ld b,d	66
	ld c,e	75
	push de	213
	ld hl, O	33 O O
	ld de, 1OOO	17 232 3
	call add	205 226* 126*
	ld de, 100	17 100 0
	call add	205 226* 126*
	ld e, 10	30 10
	call add	205 226* 126*

```
                                 ld a, (bc)            10
                                 sub 48                214  48
                                 ld e, a               95
                                 add hl,de             25
                                 ld b,h                68
                                 ld c,l                77
                                 ld hl, (23635)        42  83  92
"Find Line" (encontrar           inc hl                35
             la línea)           inc hl                35
"End of Prog" (fin del           call check            205  76*  127*
               programa)         jr c, exists          56  3
                                 pop hl                225
                                 jr search             24  153
"Exists" (existe, prueba         ld a, (hl)            126
          su existencia)         cp c                  185
                                 jr nc, next byte      48  7
                                 inc hl                35
"Wrong Line" (línea              inc hl                35
              equivocada)        call end of line      205  65*  127*
                                 jr find line          24  235
"Next Byte" (siguiente octeto)   inc hl                35
                                 ld a, (hl)            126
                                 cp b                  184
                                 jr c, wrong line      56  245
                                 dec hl                43
                                 dec hl                43
                                 ld c, (hl)            78
                                 dec hl                43
                                 ld h, (hl)            102
                                 ld l, c               105
                                 pop bc                193
                                 push bc               197
                                 push hl               229
                                 ld de, 1000           17  232  3
                                 call insert           205  212*  126*
                                 ld de, 100            17  100  0
                                 call insert           205  212*  126*
                                 ld e, 10              30  10
                                 call insert           205  212*  126*
                                 ld e, 1               30  1
                                 call insert           205  212*  126*
                                 inc bc                3
                                 sub a                 151
                                 ld (bc),a             2
                                 inc bc                3
```

	ld (bc),a	2
	inc bc	3
	pop hl	225
	ld a,l	125
	ld (bc),a	2
	inc bc	3
	ld a,h	124
	ld (bc),a	2
	inc bc	3
	sub a	151
	ld (bc),a	2
	pop hl	225
	jp search	195 15* 126*
"Restore" (volver a almacenar)	ld hl, (23635)	42 83 92
"Following" (siguiente	inc hl	35
inmediato, siguiendo)	inc hl	35
	call check	205 76* 127*
	ret nc	208
	ld b,h	68
	ld c,l	77
	call end of line	205 65* 127*
	push hl	229
	and a	167
	sbc hl,bc	237 66
	dec hl	43
	dec hl	43
	ld a,l	125
	ld (bc),a	2
	inc bc	3
	ld a,h	124
	ld (bc),a	2
	pop hl	225
	jr following line	24 231
"Insert" (insertar)	ld a, 48	62 48
"Subtract"(sustraer)	and a	167
	sbc hl,de	237 82
	jr c, poke	56 3
	inc a	60
	jr subtract	24 248
"Poke" (introducir mediante	add hl,de	25
carga)	ld (bc),a	2
	inc bc	3
	ret	201
"Add" (sumar)	ld a, (bc)	10
	inc bc	3

	sub 47	214 47
"Repeat" (repetir)	dec a	61
	ret z	200
	add hl,de	25
	jr repeat	24 251
"Find" (encontrar)	ld a, (hl)	126
	call check	205 76* 127*
	ret nc	208
	cp 234	254 234
	jr nz not REM	32 13
"Find Enter" (encontrar el carácter Enter)	inc hl	35
	ld a, (hl)	126
	cp 13	254 13
	jr nz, find ENTER	32 250
"Increase" (incrementar)	inc hl	35
	inc hl	35
	inc hl	35
	inc hl	35
	inc hl	35
	jr find	24 234
"Not REM" (no REM)	cp 34	254 34
	jr nz, not string	32 9
"Next Character" (siguiente carácter)	inc hl	35
	ld a, (hl)	126
	cp 34	254 34
	jr nz, next character	32 250
	inc hl	35
	jr find	24 221
"Not String" (que no sean cadenas)	cp 13	254 13
	jr z, increase	40 232
	call 6326	205 182 24
	jr z, find	40 212
	cp 237	254 237
	jr z, check digit	40 27
	cp 236	254 236
	jr z, check digit	40 23
	cp 247	254 247
	jr z, check digit	40 19
	cp 240	254 240
	jr z, check digit	40 15
	cp 229	254 229
	jr z, check digit	40 11
	cp 225	254 225
	jr z, check digit	40 7
	cp 202	254 202

	jr z, check digit	40 3
	inc hl	35
	jr find	24 181
"Check Digit" (comprobar el dígito)	inc hl	35
	ld a, (hl)	126
	cp 48	254 48
	jr c, find	56 175
	cp 58	254 58
	jr nc, find	48 171
	ret	201
"End of File" (fin del archivo)	ld a, (hl)	126
"Again" (de nuevo, otra vez)	call 6326	205 182 24
	jr z, again	40 251
	cp 13	254 13
	inc hl	35
	jr nz, end of line	32 245
"Check" (comprobar)	push hl	229
	push de	213
	ld de, (23627)	237 91 75 92
	and a	167
	sbc hl,de	237 82
	pop de	209
	pop hl	225
	ret	201

Funcionamiento

El registro dual *hl* se carga con el primer número de línea. Si éste es cero, la rutina vuelve al Basic. El registro *hl* se carga luego con el paso (separación entre dos líneas consecutivas del programa en Basic) y si éste es cero, la rutina vuelve al Basic.

El registro *hl* se carga con la dirección del programa Basic, el registro *de* se emplaza al número de la línea primera. La subrutina "Check" es llamada, y caso de alcanzarse el final del programa Basic, se ejecuta un salto a "Find GOTO". El registro *bc* se carga con el antiguo número de la línea hallada, y el número se reemplaza por *de,* el registro *bc* es transferido a los punteros de líneas.

El registro *hl* es guardado en el área de conservación de registros, cargado con el valor del paso e incrementado por *de*. El resultado se copia en *de,* siendo el número de la línea siguiente. El registro *hl* es recogido del área de conservación de registros, y la rutina "End of Line" lo incrementa, de forma que apunte a la siguiente línea. La rutina luego vuelve atrás al lazo "Next Line". En "Find GOTO", el registro *hl* se carga con la dirección del programa Basic y éste se incrementa para apuntar al primer carácter de la primera línea. La subrutina "Find" se llama a continuación. Si no hay más "GOTO", "GOSUB", etc. que tengan que ser alterados, la rutina salta a "Restore". De lo contrario, una vez que se vuelve de la subrutina, el registro *hl* contiene la dirección del primer dígito después de sentencias tales como "GOTO", "GO-

SUB", etc. Este se copia en *de,* y el registro *b* se emplaza a cero. El registro *b* se usa para contar el número de dígitos que hay en el siguiente número.

El registro *b* se incrementa, y el registro *hl* se incrementa para apuntar al siguiente carácter, y este carácter se carga luego en el acumulador. Si el carácter es un punto decimal, el registro *hl* se carga con *de* y la rutina vuelve a recorrer el lazo para buscar, y en su caso encontrar, la siguiente "GOTO". Si el carácter no es la expresión "Number" la rutina ejecuta el lazo "Next Digit".

El registro *hl* se incrementa para apuntar al carácter inmediato siguiente a "Number". Si este no es el carácter "dos puntos (:)" o una expresión "Enter" la rutina vuelve a "Find Next" ya que el "GOTO" que está siendo probado tiene el destino calculado. El acumulador se carga con el valor que hay en el registro *b.* Si este valor es 4, la rutina salta a "Calculate"; si éste es más de 4, se ejecuta un salto atrás a "Find Next", por cuanto que los números de líneas que sobrepasan 9999 son inválidas.

El registro *de* se guarda en el área de conservación de registros y copiado en *hl.* El acumulador se guarda luego en el área de conservación de registros, y cargado después con el código correspondiente al carácter cero. Este se inserta en la dirección que expresa el registro dual *hl* por medio de la rutina ROM en la dirección 3976. El acumulador se recoge del área de conservación de registros e incrementado. Este contendrá el nuevo número de dígitos que existen en el número de línea. El registro *de* se recoge también del área de conservación de registros, y la rutina ejecuta el lazo "Compare".

En "Calculate", la dirección que hay en el registro *de* se transfiere al registro *bc,* luegos es guardado en el área de conservación de registros. El registro *hl* se carga con cero, y el *de* se carga con 1000. Luego se llama a la subrutina "Add", con el fin de sumarle a *hl* el número de miles que contiene el número de línea que se está escrutando. Este proceso se repite para las centenas, decenas y unidades, cargando así el registro *hl* con el número de línea. El registro dual *bc* se carga con el resultado.

El registro dual *hl* se carga con la dirección del programa Basic. A continuación se llama a la subrutina "Check" y si se alcanza al área de fin de programa, la rutina toma el registro *hl* del área de conservación de registros y salta a "Search", debido a que el "GOTO" carece de destino. Si el octeto direccionado por *hl* es menor que el valor del registro *c*, el registro *hl* se incrementa para apuntar a la siguiente línea, y se efectúa un salto a "Find Eine". En caso contrario *hl* se incrementa para apuntar al lugar del octeto siguiente, perteneciente al número de línea bajo prueba. Si éste es menor que el valor del registro *b* se ejecuta un salto a "Wrong Line".

Para llegar a esta etapa, el destino del "GOTO" ha tenido que ser encontrado. El registro *hl* se decrementa para que apunte al comienzo de la línea, y luego cargado con su nuevo número de línea. El registro *bc* se carga con la dirección que tiene el área de conservación de registros para luego guardar *hl* en la citada área. El registro *bc* contiene ahora la dirección a la cual el número de línea debe ser copiado. El *de* se carga con 1000, luego se llama a la subrutina "Insert". Esta calcula el número de miles que hay en *hl,* suma 48 para producir un dígito legible y carga el valor en *bc*. El registro *bc* es incrementado para apuntar al próximo carácter. Este proceso se repite para las cententas, decenas y unidades.

A continuación, la representación binaria del número de línea se comienza a construir; el registro *bc* se incrementa para apuntar al carácter que va después de la expresión "Number" y los dos octetos siguientes son cargados con cero. El registro *hl* es recuperado del área de conservación de registros y cargado en los siguientes dos octetos. El quinto octeto del número es cargado con cero. El registro *hl* se recoge del área de conservación de registros y la rutina salta a "Search" para repetir el proceso con el próximo "GOTO".

En "Restore", el registro *hl* se carga con la dirección del área de programa en Basic, y luego incrementado dos veces para direccionar los punteros de la línea siguiente, que contiene actualmente el antiguo número de línea. La subrutina "Check" es reclamada y si se alcanza el final del programa Basic, la rutina vuelve al Basic. El registro *bc* se carga con la dirección que expresa *hl* y se llama a la subrutina "End of Line". Este vuelve a la dirección de la expresión "Enter" (que está en *hl*) más uno. El registro *hl* se guarda en el área de conservación de registros. El registro *bc* se resta del *hl,* y luego el registro *hl* se decrementa por dos veces, produciendo los nuevos punteros de líneas que van a ser cargados en *bc* y en *bc* + 1. El registro *hl* se recupera desde el área de conservación de registros y se ejecuta un salto a "Following Line".

Subrutinas

"Insert" (Insertar)

El acumulador se carga con el código del carácter cero. El registro *de* se resta del *hl* y si el resultado es negativo, se ejecuta un salto a "Poke". De lo contrario, el acumulador se incrementa y se ejecuta el loop "Subtract".

En "Poke', el registro *de* se suma al *hl* para producir un valor positivo. El registro *bc* se carga con el valor del acumulador, y luego incrementado para apuntar al siguiente octeto. Posteriormente se sale de la subrutina (por medio de "Return").

"Add" (Sumar)

El acumulador se carga con el octeto direccionado por *bc,* y *bc* se incrementa para apuntar al siguiente octeto. Se resta 47 del acumulador. El acumulador se decrementaa y si el resultado es cero se ejecuta un "Return" (vuelta desde fin de subrutina). De lo contrario, el registro *de* se suma a *hl,* y la rutina ejecuta el lazo "Repeat".

"Find" (Encontrar)

El acumulador se carga con el octeto direccionado por *hl.* Se llama luego a la subrutina "Check" y si se llega al final del programa Basic se ejecuta un "Return". Si el carácter en el acumulador no es la expresión "REM" se ejecuta un salto a "Not REM". El registro *hl* se incrementa para apuntar al primer carácter de la línea siguiente, y se ejecuta un salto a "Find".

En "Not REM", si el acumulador no contiene el código del carácter "comillas", se ejecuta un salto a "Not String". De lo contrario, *hl* se incrementa repetidamente hasta que se encuentre con un segundo símbolo "comilla". El registro *hl* se incrementa una vez más, para apuntar al próximo carácter, y se ejecuta un salto atrás a "Find".

En "Not String", si el acumulador contiene la expresión "Enter" se ejecuta el lazo "Increase". Si éste contiene la expresión "Number" la rutina va al lazo "Find". Si no se encuentra ninguna de las instrucciones "GOSUB, GOTO, RUN, LIST, RESTORES, LLIST, LINE", el registro *hl* se incrementa y se ejecuta un salto a "Find". El registro *hl* se incrementa y el acumulador se carga con el siguiente carácter. Si éste no está dentro del rango 48 - 57, la rutina salta a "Find". Luego la rutina vuelve a la rutina principal.

"End of Line" (Final de línea)

El acumulador se carga con el octeto direcionado por *hl*. Si éste es la expresión "Number" el registro *hl* se incrementa y se ejecuta un lazo a "Again". El registro *hl* se incrementa. Si el acumulador no contiene la expresión "Enter" la rutina salta a "final de línea". Se hace una prueba para ver si el final del programa Basic ha sido ya alcanzado y si esfo ocurre, la rutina vuelve al supervisor Basic.

MEMORIA QUE QUEDA

Longitud: 14
Número de variables: 0
Prueba de la suma: 1443

Operación

Devuelve la cantidad en octetos de memoria RAM sobrante.

"Llamada" (Call)

PRINT USR dirección

"Comprobación de errores" (Error Checks)

Ninguno

Comentarios

Deberá llamarse a esta rutina antes de utilizar rutinas que puedan incrementar la longitud del programa, para asegurarnos que haya suficiente memoria RAM disponible.

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, O	33 O O
	add hl, sp	57
	ld de, (23653)	237 91 101 92
	and a	167
	sbc hl,de	237 82
	ld b,h	68

ld c,l	77
ret	201

Funcionamiento

El registro dual *hl* se emplaza a cero, y la dirección del final del área de RAM disponible se suma con él (la dirección se almacena en *sp*). El registro dual *de* se carga con la dirección de comienzo del área de RAM dispobible, y se resta de *hl*. El registro *hl* se copia en el registro *bc*, la rutina vuelve al Basic.

Longitud del Programa

Longitud: 13
Número de variables: 0
Prueba de la suma: 1544

Operación

La rutina da la longitud en octetos del programa en Basic.

"Llamada" (Call)

PRINT USR dirección

"Comprobación de errores" (Error Checks)

Ninguno

Comentarios

Ninguno

Comentarios

Ninguno

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, (23627)	42 75 92
	ld de, (23635)	237 91 83 92
	and a	167
	sbc hl,de	237 82
	ld b,h	68
	ld c,l	77
	ret	201

Funcionamiento

El registro dual *hl* se carga con la dirección del área de variables, y el registro *de* se carga con la dirección del programa Basic. El registro *de* se resta de

hl para dar la longitud del programa. El registro *hl* se copia en el *bc,* y la rutina vuelve al Basic.

Dirección de Línea

Longitud: 29
Número de variables: 1
Prueba de la suma: 2351

Operación

Da la dirección del primer carácter después de la expresión "REM" en una línea especificada.

Variables

Nombre	*Longitud*	*Posición*	*Comentario*
Número de línea	2	23296	Número de línea que deberá contener "REM".

"Llamada" (Call)

LET A = USR dirección

"Comprobación de errores" (Error Checks)

Si no existe la línea especificada o no hay en ella una sentencia "REM", la rutina devolverá el valor cero.

Comentarios

Esta rutina puede usarse para encontrar la dirección en la que el código máquina deberá cargarse para ser posicionada en una sentencia "REM".

Cuando se la llama, la variable A (se puede usar cualquier variable) se emplaza con la dirección o con un cero si ocurre un error. No introducir números de línea mayores 9999.

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, (23296)	42 0 91
	ld a,h	124
	or l	181
	jr z, error	40 5
	call 6510	205 110 25
	jr z, continue	40 4
"Error" (error, introduce el valor 0)	ld bc, O	1 0 0
	ret	201
"Continue" (continuar)	inc hl	35

```
inc hl          35
inc hl          35
inc hl          35
ld a, 234       62 234
cp (hl)         190
jr nz, error    32 243
inc hl          35
ld b,h          68
ld c,l          77
ret             201
```

Funcionamiento

El registro dual *hl* se carga con el número de línea especificado. Si este número es un cero se ejecuta un salto a "Error"; en caso contrario, la rutina ROM ubicada en la dirección 6510 es llamada al regreso de esta subrutina. El registro *hl* se emplaza con la dirección de la línea. Si el indicador cero está activado, se ejecuta una salto a "Continue". Si el indicador cero no está activo es que la línea no existe, y la rutina cae a "Error", donde *bc* se carga con cero y la rutina vuelve al Basic.

Si la rutina llega a "Continue" el registro *hl* se incrementa por cuatro para apuntar a la primera instrucción que haya en la línea especificada. Si esta instrucción no tiene un código de 234, se ejecuta un salto a "Error". Si la instrucción es una "REM", el registro *hl* se incrementa para apuntar al siguiente carácter. El valor de *hl* se copia en *bc* y la rutina vuelve al Basic.

Copiar Memoria

Longitud: 33
Número de variables: 3
Prueba de la suma: 4022

Operación

Esta rutina copia un área de memoria desde una dirección a otra.

Variables

Nombre	*Longitud*	*Posición*	*Comentario*
Comienzo	2	23296	Dirección desde la que hay que copiar.
Destino	2	23298	Dirección en la que se ha de copiar.
Longitud	2	23300	Número de octetos a copiar.

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Ninguno

Comentarios

Esta rutina puede usarse para producir "imágenes animadas" por medio del siguiente método:

I) Produce la primera imagen de información.
II) Copia la información gráfica por encima del RAMTOP.
III) Repite para otras imágenes más.
IV) Vuelve a copiar las imágenes una a una en rápida sucesión.

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, (23296)	42 0 91
	ld de, (23298)	237 91 2 91
	ld bc, (23300)	237 75 4 91
	ld a,b	120
	or c	177
	ret z	200
	and a	167
	sbc hl,de	237 82
	ret z	200
	add hl,de	25
	jr c, lddr	56 3
	ldir	237 176
	ret	201
"lddr" ("Load, decrement	ex de,hl	235
and repeat") (cargar,	add hl,bc	9
decrementar y repetir)	ex de,hl	235
(Ver glosario de	add hl,bc	9
instrucciones en	dec hl	43
código máquina en	dec de	27
capítulo 3.)	lddr	237 184
	ret	201

Funcionamiento

El registro dual *hl* se carga con la dirección del primer octeto de memoria que va a copiarse, el registro *de* se carga con la dirección en la que será copiada, y el registro *bc* se carga con el número de octetos a copiar. Si el registro *bc* es cero o el registro *hl* = *de*, la rutina vuelve al Basic. Si el registro *hl* es mayor que *de*, la sección de memoria se copia usando la instrucción "ldir" (ver capítulo 3), después la rutina regresa al Basic.

Si *de* es mayor que *hl*, se añade *bc*—1 a ambos pares de registros, se copia la memoria usando la instrucción "lddr", y la rutina vuelve a BASIC.

Poner a Cero todas las Variables

Longitud: 108
Número de variables: 0
Prueba de la suma: 10717

Operación

Todas las variables numéricas se ponen a cero, todas las cadenas dimensionadas se rellenan con espacios, las cadenas no dimensionadas se emplazan con longitud cero (cadenas nulas).

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Si no hay variables en memoria la rutina regresa al Basic inmediatamente.

Comentarios

Esta rutina es útil para ayudar a depurar programas.

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, (23627)	42 75 92
"Check" (comprobar)	ld a, (hl)	126
	cp 128	254 128
	ret z	200
	ld de, 1	17 1 0
	bit 7,a	203 127
	jr nz, next bit	32 32
	bit 5,a	203 111
	jr z, 'string'	40 9
"Zero" (pone a cero)	ld b,5	6 5
"Next Byte" (siguiente octeto)	inc hl	35
	ld (hl),d	114
	djnz next byte	16 252
	add hl,de	25
	jr check	24 232
"String" (cadena)	inc hl	35
	ld c,(hl)	78
	ld (hl),d	114
	inc hl	35
	ld b,(hl)	70
	ld (hl),d	114
	inc hl	35

"Delete" (elimina)	ld a,b	120
	or c	177
	jr z, check	40 221
	push bc	197
	call 4120	205 24 16
	pop bc	193
	dec bc	11
	jr delete	24 244
"Next Bit" (siguiente bit)	bit 6,a	203 119
	jr nz, bit 5	32 45
	bit 5,a	203 111
	jr z, array	40 7
"Number" (número)	inc hl	35
	bit 7, (hl)	203 126
	jr z, number	40 251
	jr zero	24 213
"Array" (matriz)	sub a	151
"Find Lenght" (encontrar longitud)	puch af	245
	inc hl	35
	ld c, (hl)	78
	inc hl	35
	ld b, (hl)	70
	inc hl	35
	push hl	229
	ld l, (hl)	110
	ld h, d	98
	add hl,hl	41
	pop de	209
"Find Elements" (encontrar elementos)	inc de	19
	dec bc	11
	dec hl	43
	ld a,h	124
	or l	181
	jr nz, find elements	32 249
	dec bc	11
"Rub Out" (borrar)	inc de	19
	dec bc	11
	pop af	241
	push af	245
	ld (de),a	18
	ld a,b	120
	or c	177
	jr nz, rub out	32 247
	pop af	241
"Restore" (volver a almacenar)	inc de	19

	rx de,hl	235
	jr check	24 164
"Bit 5" (bit 5)	bit 5,a	203 111
	jr z, string array	40 5
	ld de, 14	17 14 0
	jr 'zero'	24 170
"String Arraw" (matriz formada con elementos del tipo "cadenas")	ld a, 32	62 32
	jr find length	24 210

Funcionamiento

El registro dual *hl* se carga con la dirección de comienzo del área de variables. El acumulador se carga con el octeto almacenado en *hl*. Si el valor de este octeto es 128 la rutina regresa al Basic, debido a que el 128 señala el final de las variables. El registro *de* se carga con el valor 1 para usarse en la rutina posteriormente. Si el bit 7 del acumulador está activado (puesto a uno) la rutina salta a "Next Byte", luego si el bit 5 está desactivado (puesto a 0), la rutina salta a "String".

Para alcanzar "zero" sin que haya saltos hacia adelante en la rutina, la variable encontrada deberá ser un número cuyo nombre tiene una letra de longitud (tenga un carácter de longitud). El registro *b* se emplaza a cinco, para usarse como contador, el registro *hl* se incrementa para apuntar al siguiente octeto y éste toma el valor cero. El contador se decrementa y si aún no es cero, la rutina vuelve al principio del lazo "Next Byte". El registro *de* se suma con *hl* para apuntar a la siguiente variable y se ejecuta un salto atrás hasta "Check".

Si la rutina alcanza la etiqueta "String", el registro *hl* se incrementa con el fin de apuntar a los octetos que contienen la longitud de la cadena hallada. La longitud antigua se carga en *bc* para usarse como contador, emplazando a cero la nueva longitud. Se incrementa de nuevo el registro *hl* para apuntar al texto de la cadena. Si el contador está emplazado a cero, *hl* apunta ahora a la siguiente variable de manera que se salta atrás hasta *"Check"*. En el caso de que no esté a cero, el registro *bc* se guarda en el área de conservación de registros y se llama a la rutina ROM cuya dirección es 4120 para eliminar un carácter. El contador se extrae del área de conservación de registros, se decrementa y se hace un salto atrás a *"Delete"*.

En "Next Bit", se comprueba el bit 6 del acumulador. Si éste está a 1 se ejecuta un salto a "Bit 5", ya que una matriz de cadena o una variable de control del tipo "For/Next" ha sido encontrada. Si éste está emplazado a cero, y el bit 5 está también a cero, se ejecuta un salto a "Array".

Para llegar a "Number", la variable encontrada debe ser un número con un nombre que tenga mayor longitud que un carácter. El registro dual *hl* se incrementa para apuntar al siguiente octeto, y esto se repite hasta que se encuentre un octeto con el bit siete emplazado a 1. Cuando éste se encuentra, la rutina salta a "zero" para cargar a la variable con cero.

Si la rutina alcanza "Array" el acumulador se carga con cero, porque éste es el valor al que deberá estár emplazado más tarde.

En “Find Lenght”, el acumulador se guarda en el área de conservación de registros y el registro *hl* se incrementa para apuntar a los octetos que contienen la longitud de la matriz. Esta se copia en *bc* para usarse como contador. El registro *hl* es incrementado de nuevo, de forma que ahora apunte al octeto que contiene el número de dimensiones, y luego *hl* se vuelve a guardar en el área de conservación de registros. *hl* se carga con el número de dimensiones y éste se multiplica por dos. El registro *de* es emplazado con la dirección guardada en el área de conservación de registros, luego se incrementa *de* tantasa veces como exprese el registro *hl* y el *bc* se decrementa (*hl* + 1) veces. El registro *de* se incrementa de nuevo y el registro *bc* se decrementa. Así el registro *de* apunta ahora al siguiente elemento de la matriz y *bc* contiene el número de octetos que quedan antes de alcanzar el final. El acumulador se extraé del área de conservación de registros y es cargado en el registro *de*. El contador que hay en *bc* se decrementa y si no contiene cero, la rutina salta a “Rub Out”. El valor en *hl* se ajusta después para apuntar a la siguiente variable, finalmente salta a “Check”.

En “Bit 5” se hace una comprobación con el fin de ver si una matriz de cadena ha sido hallada. Caso de que ocurra, el acumulador se emplaza con el código correspondiente al carácter “espacio” (carácter en blanco) y se ejecuta un salto a “Find Lenght”. Para llegar a este punto, la variable deberá ser una variable de control “For/Next”. El registro *de* se emplaza a 14, de forma que sumando éste con (*hl* + 5) apunte a la variable siguiente. La rutina salta luego a “zero”.

Listar Variables

Longitud: 94
Numero de variables: 0
Prueba de la suma: 10295

Operación

Esta rutina lista los nombres de todas las variables que en ese momento se hallen en la memoria.

“Llamada” (Call)

RAND USR dirección

“Comprobación de errores” (Error Checks)

Si no existen variables en memoria la rutina regresa inmediatamente al Basic.

Comentarios

Esta rutina es una ayuda útil en la depuración de programas, en particular con programas largos o complejos.

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	res O, (IY + 2)	253 203 2 134
	ld hl, (23627)	42 75 92
"Next Variable" (siguiente variable)	ld a, 13	62 13
	rst 16	215
	ld a, 32	62 32
	rst 16	215
	ld a, (hl)	126
	cp 128	254 128
	ret z	200
	bit 7,a	203 127
	jr z, bit 5	40 62
	bit 6,a	203 119
	jr z, next bit	40 31
	bit 5,a	203 111
	jr z, string array	40 9
	sub 128	214 128
	ld de, 19	17 19 0
"Print" (exponer)	rst 16	215
	add hl,de	25
	jr next variable	24 225
"String Array" (matriz de cadena)	sub 96	214 96
	rst 16	215
	ld a, 36	62 36
"Brackets" (paréntesis)	rst 16	215
	ld a, 40	62 40
	rst 16	215
	ld a, 41	62 41
"Pointers" (punteros)	inc hl	35
	ld e, (hl)	94
	inc hl	35
	ld d, (hl)	86
	inc hl	35
	jr print	24 234
"Next Bit" (siguiente bit)	bit 5,a	203 111
	jr z, array	40 19
	sub 64	214 64
	rst 16	215
"Next Character" (siguiente carácter)	inc hl	35
	ld a, (hl)	126
	bit 7,a	203 127
	jr nz, last character	32 3
	rst 16	215
	jr next character	24 247

"Last Character" (último carácter)	sub 128	214 128
"Jump" (saltar)	ld de, 6	17 6 0
	jr print	24 211
"Array" (matriz)	sub 32	214 32
	jr brackets	24 216
"Bit 5" (bit 5)	bit 5,a	203 111
	jr nz, jump	32 243
	add a, 32	198 32
	rst 16	215
	ld a, 36	62 36
	jr pointers	24 211

Funcionamiento

El bit 0 del octeto cuya dirección es 23612 se desactiva para asegurar que cualesquiera que sean los caracteres impresos, aparezcan en la parte superior de la pantalla. El registro *hl* se carga con la dirección del área de variables. El acumulador se carga con la expresión "Enter" y ésta se expone usando la rutina ROM que comienza en la dirección 16. Luego el acumulador se carga con el código del carácter "espacio" (carácter en blanco) y éste se expone usando la misma rutina.

El acumulador se carga con el octeto almacenado en la dirección expresada en el registro *hl.* Si el valor de éste es 128 la rutina regresa al Basic debido a que se ha llegado al final del área de variables.

Si el bit 7 del acumulador está emplazado a cero, la rutina salta a "Bit 5" porque ha sido hallada una caden, o un número cuyo nombre tiene la longitud de un carácter sólmante (1 letra). El bit 6 del acumulador se prueba. Si éste está emplazado a cero se salta a "Next Bit" debido a que una matriz, o en su caso un número cuyo nombre es mayor que un carácter (más de una letra), fue hallada. Si el bit 5 del acumulador es cero, la rutina salta a "String Array".

La rutina llega a este punto si la variable hallada es la de control de un lazo "For/Next", 128 se resta del acumulador, resultando de esto el código del carácter a exponer. El registro *de* se carga con 19 para apuntar a la siguiente variable cuando se sume con *hl,* el carácter del acumulador se expone, el registro *de* se suma a *hl,* la rutina retrocede al lazo "Next Variable".

Si la rutina alcanza "String Array", se resta del acumulador el valor 96 para dar el código del nombre de la matriz hallada. Este se expone usando la rutina ROM (cuya dirección está en ROM). Un signo de dólar ($) y un paréntesis abierto (*(*) son expuestas, luego el acumulador se carga con el código de un paréntesis cerrado (*)*). El registro *hl* se incrementa para apuntar a los octetos que contienen la longitud de la matriz. Estos se cargan en el registro *de,* de forma que sumándolo al registro *hl* nos dé la dirección de la siguiente variable. Se ejecuta un salto a "Print", donde se expone el paréntesis cerrado y el registro *de* se suma al *hl.*

En "Next Bit" se comprueba el bit 5 del acumulador. Si éste está a cero, se ejecuta un salto a "Array". Si está a uno, es que se ha encontrado con un número cuyo nombre es más largo que un carácter (una letra). Al acumulador se le resta el valor 64 y el carácter resultante se expone. Luego la rutina ejecuta repetidos lazos, exponiendo cada carácter que encuentre, hasta que halle uno cuyo bit 7 esté emplazado a uno. El valor 128 se resta del código de carácter, *de* se carga con el desplazamiento hasta la siguiente variable, y la rutina salta a "Print".

Si se encuentra una matriz, el valor 32 se resta del acumulador para dar el código correcto, y se ejecuta un salto a "Brackets".

En "Bit 5", si se encuentra un número cuyo nombre tenga una letra sólo, la rutina vuelve al lazo "Jump".

Para llegar a este punto, la variable hallada deberá ser una cadena. Restando 32 del acumulador nos da el código que será expuesto. Finalmente, el acumulador se carga con el código del signo dólar, y se ejecuta un salto a "Pointers".

Buscar y listar

Longitud: 155
Número de variables: 2
Prueba de la suma: 17221

Operación

Esta rutina busca a través de todo el programa en Basic y lista cada línea que contenga una cadena de caracteres especificados por el usuario.

Variables

Nombre	*Longitud*	*Posición*	*Comentarios*
Comienzo de datos	2	23296	Dirección del primer octeto de datos.
Longitud de la cadena	1	23298	Número de caracteres que hay en la cadena.

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Si no existe en memoria un programa Basic o la longitud de la cadena de caracteres es cero, entonces la rutina regresa inmediatamente al Basic.

Comentarios

El tiempo empleado por esta rutina es proporcional a dos elementos; la longitud de la cadena y la longitud del programa Basic.

La cadena que hay que buscar deberá estar introducida (POKE) por encima del "RAMTOP" y la dirección del primer octeto de la cadena deberá introducirse en las posiciones 23296 y 23297. La longitud de la cadena deberá almacenarse en 23298.

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	res O, (IY + 2)	253 203 2 134
	ld ix, (23296)	221 42 0 91
	ld hl, (23635)	42 83 92
"Restart" (volver a comenzar)	ld a, (23298)	58 2 91
	ld e, a	95
	cp O	254 0
	ret z	200
	push hl	229
"Restore" (volver a almacenar)	push ix	221 229
	pop bc	193
	ld d, O	22 0
	inc hl	35
	inc hl	35
	inc hl	35
"Check" (comprobar)	inc hl	35
	push de	213
	ld de, (23627)	237 91 75 92
	and a	167
	sbc hl,de	237 82
	add hl,de	25
	pop de	209
	jr c, enter	56 4
	pop hl	225
	ret	201
"Long Jump" (salto relativamente largo)	jr restart	24 223
"Enter" (introducir valor o línea)	ld a, (hl)	126
	cp 13	254 13
	jr nz, number	32 5
	inc hl	35
	pop bc	193
	push hl	229
	jr restore	24 221
"Number" (número)	call 6326	205 182 24
	jr nz, compare	32 8
	dec hl	43
"Different" (es diferente?)	push ix	221 229

	pop bc	193
	ld d, O	22 0
	jr check	24 216
"Compare" (comparar)	ld a, (bc)	10
	cp (hl)	190
	jr nz, different	32 245
	inc bc	3
	inc d	20
	ld a,d	122
	cp e	187
	jr nz, check	32 206
	ld a, 13	62 13
	rst 16	215
	pop hl	225
	push hl	229
	ld b, (hl)	70
	inc hl	35
	ld l, (hl)	110
	ld h, b	96
	ld de, 1000	17 232 3
	ld a, 47	62 47
"Thousands" (miles)	inc a	60
	and a	167
	sbc hl,de	237 82
	jr nc, thousands	48 250
	add hl, de	25
	rst 16	215
	ld de, 100	17 100 0
	ld a, 47	62 47
"Hundres" (centenas)	inc a	60
	and a	167
	sbc hl, de	237 82
	jr nc, hundreds	48 250
	add hl,de	25
	rst 16	215
	ld de, 10	17 10 0
	ld a, 47	62 47
"Tens" (decenas)	inc a	60
	and a	167
	sbc hl,de	237 82
	jr nc, tens	48 250
	add hl, de	25
	rst 16	215
	ld a,l	125
	add a, 48	198 48
	rst 16	215

	pop hl	225
	inc hl	35
	inc hl	35
	inc hl	35
"Next Character" (siguiente carácter)	inc hl	35
	ld a, (hl)	126
"Line End" (fin de línea)	cp 13	254 13
	jr nz, chr 14	32 4
	rst 16	215
	inc hl	35
	jr long jump	24 155
"Chr 14" (carácter 14)	call 6326	205 182 24
	jr z, line end	40 243
	cp 32	254 32
	jr c, next character	56 237
	rst 16	215
	jr next character	24 234

Funcionamiento

El bit 0 del octeto almacenado en la dirección 23621 se desactiva para asegurar que cualesquiera carácter expuestos, aparezcan en la parte superior de la pantalla. El registro índice *ix* se carga con la dirección del primer octeto de datos. Esto le permite a la dirección ser cargada en otros registros duales, usando así menos referencias a la memoria intermedia de impresora. El registro *hl* se carga con la dirección del programa en Basic.

El acumulador se carga con la longitud de la cadena y se copia en el registro *e*. Si la longitud es cero, la rutina regresa al Basic inmediatamente. La dirección en *hl* se guarda en el área de conservación de registros, conteniendo la posición en memoria de la línea que se está buscando.

La dirección de los datos se transfiere desde *ix* al registro *bc* con el fin de ser más accesible. El registro *d* se carga con cero, es decir, que el número de caracteres encontrado se iguale al de los datos introducidos. El registro dual *hl* se incrementa por tres para apuntar al octeto superior o alto del puntero de línea. El registro *hl* se incrementa para apuntar al siguiente carácter. El registro *de* se guarda en el área de conservación de registros.

El registro *de* se carga con la dirección del área de variables, y ésta se resta de *hl*. Si el resultado es negativo, la rutina salta a "Enter" después de volver a almacenar el registro *hl* y extraer *de* desde el área de conservación de registro. Si el resultado fue positivo, el área antes citada es restablecida a su tamaño original y la rutina vuelve al Basic, ya que se ha llegado al final del programa en Basic.

En "Enter" el acumulador se carga con el octeto almacenado en la dirección expresada en *hl*. Si ésta no es la expresión "Enter" se ejecuta un salto a "Number". Si la expresión "Enter" es hallada, se incrementa el registro *hl* para apuntar al comienzo de la línea siguiente. La dirección de la línea anterior se remueve del área de conservación de registros y se reemplaza por el nuevo valor que hay en *hl*. Luego se ejecuta un salto a "Restore".

En "Number", se llama a la rutina ROM cuya dirección es 6326. Si el carácter que hay en el acumulador es la expresión "Number", se incrementa *hl* para apuntar al primer carácter, detrás de la rutina ROM, que se encuentra por medio de la representacińn binaria. Si la expresión "Number" no se encuentra, la rutina salta a "Compare"; de lo contrario se decrementa *hl* y la rutina atraviesa hasta "Different". El registro *bc* se copia del registro ix, el número de caracteres encontrado se desactiva (se pone a cero), y se ejecuta un salto a "Check".

En "Compare" el acumulador se carga con el octeto almacenado en la dirección expresada en *bc*. Si éste resulta no ser el mismo que el octeto almacenado en la dirección expresada en *hl*, la rutina vuelve al lazo "Different". El registro *bc* se incrementa para apuntar al siguiente octeto de datos, y el número de caracteres encontrado se incrementa. Si no es igual a la longitud de la cadena, la rutina retrocede hasta "Check".

El acumulador se carga con el código de "Enter" (código de la expresión) y éste se expone usando la rutina ROM en la dirección 16. La dirección de la línea a exponer se carga, extrayéndola del área de conservación de registros (Stack), en el registro *hl*. El número de línea se copia luego en *hl* vía el registro *b*. El registro *de* se carga con 1000 y el acumulador se carga con el código del carácter "0" (cero) menos uno. El acumulador se incrementa y el registro *de* se resta repetidamente del registro *hl* hasta que *hl* sea negativo. Luego, el registro *de* se suma una sola vez con el *hl* para producir un resto positivo. El carácter que hay en el acumulador se expone a continuación.

El proceso anterior es repetido para valores del registro *de* = 100 y *de* = 10. Luego, se carga el resto en el acumulador, se le suma 48, y el carácter resultante se expone.

La dirección de comienzo de la línea es extraída del área de conservación de registros y cargada en *hl*. Luego se incrementa *hl* para apuntar al octeto superior del puntero de la línea, se incrementa *hl*, y el octeto en *hl* se carga en el acumulador. Si este octeto no es la expresión "Enter" se efectúa un salto a "Chr 14"; en caso contrario, se imprime la expresión "Enter", se incrementa *hl* y se ejecuta un salto atrás a "Restart".

En "Chr 14", se llama a la rutina ROM ubicada en la posición 6326. Si el carácter que hay en el acumulador es la expresión "Number", se incrementa el registro *hl* para apuntar al primer carácter detrás del número encontrado; ese carácter se carga en el acumulador, luego se ejecuta un salto a "Line End". A continuación, si el carácter que hay en el acumulador tiene un código menor que 32, la rutina vuelve al lazo "Next Character". Si el código es más que 31, el carácter encontrado se expone y se ejecuta un salto a "Next Character".

Buscar y reemplazar

Longitud: 85

Número de variables: 3

Prueba de la suma: 8518

Operación

Esta rutina busca, dentro de un programa Basic, una cadena de caracteres y reemplaza ésta cada vez que la encuentra, por otra cadena de la misma longitud.

Variables

Nombre	*Longitud*	*Posición*	*Comentario*
Antiguo comienzo datos	2	23296	Dirección de la cadena que será reemplazada.
Longitud de la cadena	1	23298	Longitud de la cadena que será reemplazada.
Comienzo de nuevos datos	2	23299	Dirección de la cadena de reemplazamiento.

"Llamada" (Call)

RAND USR dirección

"Comprobación de errores" (Error Checks)

Si la longitud de la cadena es cero o no existe programa en Basic en la memoria, la rutina regresa al Basic inmediatamente.

Comentarios

El tiempo empleado por esta rutina depende de la longitud de la cadena y de la longitud del programa en Basic que hay en memoria.

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld ix, (23296)	221 42 0 91
	ld hl, (23635)	42 83 92
	ld a, (23298)	58 2 91
	ld e, a	95
	cp O	254 O
	ret z	200
	dec hl	43
"New Line" (nueva línea)	inc hl	35
	inc hl	35
	inc hl	35
	inc hl	35
	jr reset	24 23
"Check" (comprobar)	inc hl	35
	push de	213
	ld de, (23627)	237 91 75 92
	and a	167
	sbc hl,de	237 82

	add hl,de	25
	pop de	209
	ret nc	208
	ld a, (hl)	126
	cp 13	254 13
	jr z, new line	40 233
	call 6326	205 182 24
	jr nz, compare	32 8
	dec hl	43
"Reset" (desactivar)	push ix	221 229
	pop bc	193
	ld d, O	22 0
	jr check	24 226
"Compare" (comparar)	ld a, (bc)	10
	cp (hl)	190
	jr nz, reset	32 245
	inc bc	3
	inc d	20
	ld a,d	122
	cp e	187
	jr nz, check	32 216
	push hl	229
	ld d, O	22 0
	and a	167
	sbc hl,de	237 82
	ld d,e	83
	ld bc, (23299)	237 75 3 91
	inc d	20
"Next Character" (siguiente	inc hl	35
carácter)	dec d	21
	jr z, finish	40 5
	ld a, (bc)	10
	ld (hl),a	119
	inc bc	3
	jr next char	24 247
"Finish" (terminar)	pop hl	225
	jr reset	24 215

Funcionamiento

El registro índice *ix* se carga con la dirección de la cadena que se está buscando. Esta deberá estar por encima del "RAMTOP". El registro *hl* se carga con la dirección del àrea de programa, y el acumulador se carga con la longitud de la cadena. La longitud se copia en el registro *e* para usarse más tarde en el programa. Si la longitud de la cadena es cero, la rutina vuelve al Basic. El registro *hl* se ajusta para apuntar al octeto superior del siguiente puntero de línea Basic y se ejecuta un salto a "Reset".

En "Check" se incrementa el registro *hl* para apuntar al siguiente carácter. El registro *de* se guarda en el área de conservación de registros y se carga con la dirección del área de variables. Si el registro *hl* no es menor que *de* es que se ha llegado al final del programa, y después de volver a almacenar el registro *de* en el área de conservación de registros, la rutina vuelve al Basic.

El acumulador se carga con el carácter direccionado por el registro *hl.* Si este es la expresión "Enter", la rutina vuelve atrás hasta el lazo "New Line". Si el acumulador no contiene la expresión "Number" (carácter 14), se ejecuta un salto a "Compare"; en caso contrario se incrementa el registro por cinco (cinco veces), de manera que *hl* apunte al quinto octeto del número hallado.

En "Reset", el registro *bc* se carga con la dirección de la cadena que se está buscando. El registro *d* se emplaza a cero para contener el número de caracteres que tenga la cadena hallada. La rutina vuelve al lazo "Check".

En "Compare" el acumulador se carga con el carácter que haya en la cadena y es precisamente el que va a estar apuntando por el registro *bc.* Si este es diferente al octeto direccionado por *hl,* la rutina salta a "Reset". El registro *bc* se incrementa para apuntar al siguiente carácter que hay en la cadena, el contador que hay en el registro *d* se incrementa. Si éste no es igual a la longitud de la cadena la rutina vuelve al lazo "Check".

Si se encuentra la cadena, el registro *hl* se guarda en el área de conservación de registros, de forma que la rutina comience buscando desde esta dirección la siguiente aparición de la cadena. El registro *de* se carga con la longitud de la cadena y ésta se resta de *hl* para dar la dirección de comienzo menos uno. La longitud de la cadena es cargada luego en el registro *d* para usarse como contador. El registro *bc* se carga con la dirección de comienzo de la nueva cadena, y el registro *d* se incrementa. El registro *hl* se incrementa para apuntar a la siguiente posición y el contador se decrementa. Si el contador contiene cero, el registro *hl* se extrae del área de conservación de registros y se ejecuta un salto a "Reset" para encontrar la siguiente aparición de la cadena. El acumulador se carga con el carácter apuntado por *bc* y éste es cargado en *hl.* El registro *bc* se incrementa para apuntar al siguiente carácter y la rutina vuelve atrás hasta el lazo "Next Char".

Buscar en ROM (Subrutinas)

Longitud: 58
Numero de variables: 3
Prueba de la suma: 6533

Operación

Esta rutina busca en la ROM cierto modelo (o configuración) de octetos especificados por el usuario.

Variables

Nombre	*Longitud*	*Posición*	*Comentario*
Comienzo de búsqueda	2	23296	Comienzo de la memoria en que se quiere buscar.
Longitud de la cadena	1	23298	Número de octetos en la cadena.
Comienzo de los datos	2	23299	Dirección de la cadena en la RAM.

"Llamada" (Call)

PRIN USR dirección

"Comprobación de errores" (Error Checks)

Si es cero la longitud de la cadena, la rutina regresa al Basic de inmediato, dando la dirección de comienzo de los datos. Si no encuentra la cadena, devuelve el valor 65535.

Comentarios

Cuando se escriben programas en código máquina, esta rutina puede usarse para encontrar subrutinas en la ROM, si el usuario conoce ya cómo está escrita esa parte de la rutina.

Como la mayor parte de la ROM del ZX Spectrum es una adaptación de la del ZX 81, los programas escritos originalmente para el ZX 81 se pueden adaptar fácilmente. Por ejemplo, la rutina "dirección de línea" llama a la rutina en ROM cuya dirección es 6510. En el ZX 81 la rutina comienza en la dirección 2520. Desensamblando esta rutina, tenemos:

```
push hl
ld hl, program
ld d,h*
ld e,l*
pop bc*
call 09EA
```

Los tres octetos señalados con un asterisco son los mismos en el Spectrum, y se pueden encontrar usando la rutina de búsqueda. De hecho, la rutina devuelve la dirección 6514, lo cual nos indica que se ha añadido un 4 a la dirección de comienzo de la requerida rutina ROM.

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	ld hl, (23296)	42 0 91
	ld de, (23298)	237 91 2 91
"Restart" (volver a comenzar)	ld bc, (23299)	237 75 3 91
	ld a,e	123
	cp O	254 O
	ret z	200

	push hl	229
	ld d, O	22 O
"Compare" (comparar)	ld a, (bc)	10
	cp (hl)	190
	jr z, match	40 25
	pop hl	225
	inc hl	35
	push de	213
	push hl	229
	ld hl, 16384	33 0 64
	l d, O	22 0
	and a	167
	sbc hl,de	237 82
	inc hl	35
	pop de	209
	and a	167
	sbc hl,de	237 82
	ex de,hl	235
	pop de	209
	jr nz, restart	32 220
	ld bc, 65535	1 255 255
	ret	201
"Match" (concuerda)	inc d	20
	ld a,d	122
	cp e	187
	jr nz, next byte	32 2
	pop bc	193
	ret	201
"Next Byte" (siguiente octeto)	inc hl	35
	inc bc	3
	jr compare	24 216

Funcionamiento

El registro *hl* se carga con la dirección de la primera posición en memoria que será comprobada. Para encontrar la primera aparición en ROM, éste (*hl*) deberá estar emplazado a cero. El registro *e* se carga con el número de octetos de la cadena a buscar. El registro dual *bc* se carga con la dirección de la cadena, introducida por el usuario en la RAM. El acumulador se carga con la longitud de la cadena, y si ésta es cero, la rutina vuelve al Basic.

La dirección en *hl* se guarda en el área de conservación de registros. El acumulador se carga con el octeto apuntado por el registro *bc*. Si éste resulta ser el mismo que el octeto apuntado por *hl* se ejecuta un salto a "Match". Si los dos octetos son diferentes, el registro *hl* se carga con la dirección que hay en el área de conservación de registros. Este se incrementa luego para apuntar a la posición siguiente en memoria.

Los registros *de* y *hl* se guardan en el área de conservación de registros, el registro *hl* se carga con la dirección del primer octeto de RAM, y el registro *de*

se carga con la longitud de la cadena. El registro *de* se resta del *hl* para dar la dirección de comienzo más alta posible para la cadena. Este se incrementa para apuntar a la primera dirección que no podría contener a la cadena.

La dirección en lo alto del área de conservación de registros se carga en *de* y éste se resta de *hl*. El resultado de esta operación se retiene mientras el registro *hl* se carga con el contenido del registro *de*, y el registro *de* se carga con el número en el área de conservación de registros. Si el resultado era cero, el registro *bc* se carga con 65535 y la rutina vuelve al Basic dado que no existe esa cadena en la ROM. Si el resultado no era cero, la rutina regresa al lazo "Restart".

En "Match" el registro *d* se incrementa para contener el número de octetos encontrados que han confrontado (y por lo tanto, igualados). Si éste es igual a la longitud de la cadena, el registro *bc* se extrae del área de conservación de registros y la rutina regresa al Basic. Si el registro d no contenía la longitud de la cadena, los registros *hl* y *bc* son incrementados (ambos) para apuntar a los siguientes octetos y la rutina se dirige al lazo "Compare".

INSTR$

Longitud: 168
Número de variables: 0
Prueba de la suma: 19875

Operación

Esta rutina o bien devuelve la posición de una subcadena (B$) en una cadena principal (A$) o da cero si ocurre un error.

"Llamada" (Call)

LET P = USR dirección

"Comprobación de errores" (Error Checks)

Si alguna de las cadenas no existe, o la longitud de la subcadena es cero, o la longitud de la subcadena es mayor que la longitud de la cadena principal, la rutina devuelve el valor 0.

Si no ocurre un error, pero la subcadena no se puede encontrar en la cadena principal, la rutina también devuelve un cero.

Comentarios

Cuando ocurre un "regreso" (Return) desde una rutina en código máquina, la variable P (se puede usar otra variable) contendrá valor del "regreso". Las cadenas referidas no pueden ser dimensionadas (DIM) como matrices de caracteres. Para cambiar las cadenas usadas, los números afectados de asteriscos deberán alterarse. El 66* es la subcadena, el 65* es la cadena principal. Para alterarlas, reemplazar los números por los códigos de los caracteres requeridos (A-Z = 65-90).

Listado del Código Máquina

Etiqueta	*Lenguaje ensamblador*	*Números a introducir*
	sub a	151
	ld b,a	71
	ld c,a	79
	ld d, a	87
	ld e,a	95
	ld hl, (23627)	42 75 92
"Next Variable" (siguiente variable)	ld a, (hl)	126
	cp 128	254 128
	jr z, not found	40 95
	bit 7,a	203 127
	jr nz, for-next	32 41
	cp 96	254 96
	jr nc, number	48 29
	cp 65	254 65*
	jr nz, substring	32 2
	ld d,h	84
	ld e,l	93
"Substring" (subcadena)	cp 66	254 66*
	jr nz, check	32 2
	ld b,h	68
	ld c,l	77
"Check" (comprobar)	ld a,d	122
	or e	179
	jr z, string	40 4
	ld a, b	120
	or c	177
	jr nz, found	32 38
"String" (cadena)	push de	213
	inc hl	35
	ld e, (hl)	94
	inc hl	35
	ld d, (hl)	86
"Add" (sumar)	add hl,de	25
	pop de	209
	jr increase	24 5
"Number" (número)	inc hl	35
	inc hl	35
	inc hl	35
	inc hl	35
	inc hl	35
"Increase" (incrementar)	inc hl	35
	jr next variable	24 206

"For Next" (lazo, Con/Otra)	cp 224	254 224
	jr c, next bit	56 6
	push de	213
	ld de, 18	17 18 0
	jr add	24 234
"Next Bit" (siguiente bit)	bit 5,a	203 111
	jr z, string	40 225
"Next Byte" (siguiente octeto)	inc hl	35
	bit 7, (hl)	203 126
	jr z, next byte	40 251
	jr number	24 227
"Found" (encontrado)	ex de,hl	235
	inc hl	35
	inc hl	35
	push hl	229
	push hl	229
	inc bc	3
	push bc	197
	ld a, (bc)	10
	ld e,a	95
	inc bc	3
	ld a, (bc)	10
	ld d,a	87
	or e	179
	jr z, zero length	40 11
	push de	213
	ld a, (hl)	126
	dec hl	43
	ld l, (hl)	110
	ld h,a	103
	and a	167
	sbc hl,de	237 82
	jr nc, continue	48 8
	pop bc	193
"Zero Length" (longitud cero)	pop bc	193
	pop bc	193
"Error" (error)	pop bc	193
"Not Found" (no hallado)	ld bc,	1 0 0
	ret	201
"Continue" (continuar)	pop ix	221 225
	pop bc	193
	ex de,hl	235
	pop hl	225
	inc bc	3
	inc bc	3

"Save" (guardar)	inc hl	35
	push hl	229
	push bc	197
	push ix	221 229
	push de	213
"Compare" (comparar)	ld a, (bc)	10
	cp (hl)	190
	jr z, match	40 12
	pop de	209
	pop ix	221 225
	pop bc	193
	pop hl	225
	ld a,d	122
	or e	179
	jr z, error	40 225
	dec de	27
	jr save	24 234
"Match" (concordar)	inc hl	35
	inc bc	3
	push hl	229
	dec ix	221 43
	push ix	221 229
	pop hl	225
	ld a,h	124
	or l	181
	pop hl	225
	jr nz, compare	32 227
	pop de	209
	pop de	209
	and a	167
	sbc hl,de	237 82
	pop de	209
	pop de	209
	pop de	209
	and a	167
	sbc hl,de	237 82
	ld b,h	68
	ld c,l	77
	ret	201

Funcionamiento

El acumulador, los registros duales *bc* y *de* son todos cargados con el valor cero. Posteriormente, en la rutina, el registro *bc* deberá emplazarse a la dirección de B$ y el registro *de* deberá emplazarse en la dirección de A$. El registro *hl* se carga con la dirección del área de variables.

El acumulador se carga con el octeto direccionado por *hl*. Si el acumulador contiene 128, la rutina salta a "Not Found", ya que se ha llegado al final del área de variables. Si el Bit 7 del acumulador está emplazado a uno, se ejecuta un salto a "For-Next", dado que la variable hallada no es una cadena o un número cuyo nombre tenga sólo una letra (un carácter). Si el acumulador contiene un número mayor que 95, se ejecuta un salto a "Number".

Para llegar a esta etapa, ya se ha debido de encontrar la cadena. Si el acumulador contien 65, A$ ha sido localizada y el contenido de *hl* se copia en *de*. Si el acumulador contiene 66, B$ ha sido encontrada y *hl* se copia en *bc*. Si *de* no contiene un cero y *bc* tampoco, es que se han localizado ambas cadenas, y de esta forma la rutina salta a "Found".

Si la rutina llega a "String", el registro *de* se guarda en el área de conservación de registros, luego se carga con la longitud de la cadena encontrada. Este se suma a la dirección del octeto superior de los punteros de cadena y almacenado en *hl*. El registro *de* se extrae del área de conservación de registros (Stack o Apilamiento) y se ejecuta un salto a "Increase".

En "Number", el registro *hl* se incrementa cinco veces para apuntar al último octeto de cualquier número hallado. *hl* se incrementa para apuntar a la siguiente variable y se ejecuta un salto a "Next Variable".

En "For-Next", si el acumulador contiene un número por debajo de 224, se ejecuta un salto a "Next Bit" en cuanto que la variable encontrada no es un lazo de control "For-Next". Si el valor que hay en el acumulador es más de 223, se suma 18 a *hl* para apuntar al último octeto de control y la rutina ejecuta el lazo "Increase".

Si la rutina alcanza "Next Bit", y el bit 5 del acumulador está emplazado a cero, se ejecuta un salto a "String" para cargar *hl* con la dirección de la variable siguiente ya que ha sido encontrada una matriz.

Si la rutina alcanza "Next Byte" es que se ha encontrado un número cuyo nombre tiene una longitud mayor que un carácter. Así, *hl* se incrementa hasta que apunte al último carácter del nombre de la variable y luego ejecuta un salto a "Number".

En "Found", el registro *hl* se carga con la dirección de A$, y se incrementa dos veces para dar la dirección del octeto superior de los punteros. Este valor se guarda dos veces en el área de conservación de registros. *bc* se incrementa para apuntar al octeto inferior de los punteros para B$. La dirección en *bc* se guarda luego en el área de conservación de registros. El registro *de* se carga con la longitud de B$, y si ésta es cero se ejecuta un salto a "Zero Length". El registro *de* es empujado (Push) al área de conservación de registros. El registro *hl* se carga con la longitud de A$, y si ésta no es menor que *de*, la rutina salta a "Continue". El área de conservación de registros es vuelta a su tamaño original, el registro *bc* se carga con cero, y la rutina regresa al Basic.

En "Continue", el registro índice *ix* se carga con la longitud de B$, *bc* con la dirección del octeto inferior de los punteros para B$. El registro *de* se carga con la diferencia de longitud de A$ y B$, luego *hl* se carga con la dirección del octeto superior de los punteros para A$. El registro *bc* se incrementa dos veces

para dar la dirección del primer carácter que hay en B$. El registro *hl* se incrementa para apuntar al siguiente carácter de A$. Luego, los registros duales *hl, bc, ix* y *de* son guardados en el área de conservación de registros. El acumulador se carga con el octeto direccionado por *bc,* y si éste es el mismo que el octeto direccionado por *hl* se ejecuta un salto a "Match". Los registros *de, ix, bc,* y *hl* son recuperados del área de conservación de registros. Si el registro *de* contiene cero, se ejecuta un salto a "Error" en cuanto que B$ no aparece en A$. El contador que hay en *de* es decrementado luego, y la rutina ejecuta el lazo "Save".

Si la rutina alcanza "Match", se incrementan los registros *hl* y *bc* para apuntar a los siguientes carácteres de A$ y B$ respectivamente. El registro *hl* se guarda en el área de conservación de registros. El contador que hay en *ix* se decrementa y, después de extraer *hl* del apilamiento (Stack), si *ix* no contiene cero, se ejecuta un salto atrás hasta "Compare'.

Para alcanzar esta etapa, ha tenido que aparecer B$ en A$. La longitud de B$ se sustrae de *hl.* Luego, la dirección del octeto superior de los punteros para A$ se resta de *hl.* El resultado es la posición en A$ de B$. Este se copia en el registro *bc* y la rutina regresa al Basic.

APENDICE A

Existen en este apéndice dos tablas principales de instrucciones. La tabla A2 lista las instrucciones de un octeto y las instrucciones de dos octetos que van precedidas por 203 (en hexadecimal = CB) o 237 (en hexadecimal = ED). La tabla A3. lista las instrucciones de registro índice (Index Register Instructions).

Existen muchas configuraciones en el "juego de instrucciones característico" (Instruction Set). Por ejemplo, los registros están casi siempre en el orden *b, c, d, e, h, l, (hl), a,* como por ejemplo en el grupo de instrucciones de carga de 8 bit registro a registro, números 64 a 127. De modo similar, los códigos de los registros índice emulan los códigos *hl,* estando precedidos por 221 (hexadecimal DD) cuando se refiere al registro índice *ix* y por 233 (hexadecimal FD) cuando está referido al registro índice *iy.* Algunas de las instrucciones están calificadas por uno o más caracteres que a continuación se detallan:

n Un entero de 1 octeto entre 0 y 255 inclusive.

d un desplazamiento de un octeto entre 0 y 255 inclusive (instrucciones de registro índice) o entre —127 y 128 (instrucciones de salto). Los valores negativos de *d* se obtienen sustrayendo el valor positivo de 256.

nn Un entero de dos octetos entre 0 y 65535 inclusive. El octeto más significativo está situado en segundo lugar. Por ejemplo 16384 (= 0 + 256 * 64) está contenido como 0,64.

Los calificadores están siempre situados en el octeto u octetos siguientes a la instrucción a la que se refieren, excepto en las "instrucciones de registro índice" de tres octetos (columnas 5 y 6 de la tabla A.3) en cuyo caso están situados entre el segundo y tercer octeto. Ver tabla A1 para más ejemplos.

Tabla A1. *Ejemplos del "juego de instrucciones característico del Z80A. La columna uno se refiere a las correspondientes en A2 y A3.*

Tabla y n.º de columna	*Formato general*	*Ejemplo específico*	*Decimal*			
A2,3	—	inc b	4			
A2,3	ld e,n	ld e,25	30	25		
A2,3	ld a, (nn)	ld a, (23296)	58	0	91	
A2,4	—	res 2,d	203	92		
A2,5	ld (nn),de	ld (23760),de	237	53	208	92
A3,3	—	add ix,bc	221	9		
A3,3	ld (ix + d),n	ld (ix + 193),5	221	54	193	5
A3,4	—	add iy,bc	253	9		
A3,4	ld (nn), iy	ld (23760),iy	253	34	208	92
A3,5	rrc (ix + d)	rrc (ix + 5)	221	203	5	14
A3,6	rrc (iy + d)	rrc (iy + 5)	253	203	5	14

Tabla A2. *Instrucciones del Z80A excepto para los códigos correspondientes a los registros índice (ver tabla A3.)*

Decimal	*Hexa-decimal*	*Op Code (Código de operación)*	*Después de 203 (Hex CB)*	*Después de 237 (Hex ED)*
0	00	nop	rlc b	
1	01	ld bc,nn	rlc c	
2	02	ld (bc),a	rlc d	
3	03	inc bc	rlc e	
4	04	inc b	rlc h	
5	05	dec b	rlc l	
6	06	ld b,n	rlc (hl)	
7	07	rlca	rlc a	
8	08	ex,af,af′	rrc b	
9	09	add hl,bc	rrc c	
10	0A	ld a, (bc)	rrc d	
11	0B	dec bc	rrc e	
12	0C	inc c	rrc h	
13	0D	dec c	rrc l	
14	0E	ld c,n	rrc (hl)	
15	0F	rrca	rrc a	
16	10	djnz d	rl b	
17	11	ld de,nn	rl c	
18	12	ld (de),a	rl d	
19	13	inc de	rl e	
20	14	inc d	rl h	
21	15	dec d	rl l	
22	16	ld d,n	rl (hl)	
23	17	rla	rl a	
24	18	jr d	rr b	
25	19	add hl,de	rrc	
26	1A	ld a, (de)	rr d	
27	1B	dec de	rr e	
28	1C	inc e	rr h	
29	1D	dec e	rr l	
30	1E	ld e,d	rr (hl)	
31	1F	rra	rr a	
32	20	jr nz,d	sla b	
33	21	ld hl,nn	sla c	
34	22	ld (nn),hl	sla d	
35	23	inc hl	sla e	
36	24	inc h	sla h	
37	25	dec h	sla l	
38	26	ld h,n	sla (hl)	
39	27	daa	sla a	

40	28	jr z,d	sra b	
41	29	add hl,hl	sra c	
42	2A	ld hl, (nn)	sra d	
43	2B	dec hl	sra e	
44	2C	inc l	sra h	
45	2D	dec l	sra l	
46	2E	ld l,n	sra (hl)	
47	2F	cpl	sra a	
48	30	jr nc,d		
49	31	ld sp,nn		
50	32	ld (nn),a		
51	33	inc sp		
52	34	inc (hl)		
53	35	dec (hl)		
54	36	ld (hl),n		
55	37	scf		
56	38	jr c,d	srl b	
57	39	add hl,sp	srl c	
58	3A	la d, (nn)	srl d	
59	3B	dec sp	srl e	
60	3C	inc a	srl h	
61	3D	dec a	srl l	
62	3E	ld a,n	srl (hl)	
63	3F	ccf	srl a	
64	40	ld b,b	bit 0,b	in b, (c)
65	41	ld b,c	bit 0,c	out (c),b
66	42	ld b,d	bit 0,d	sbc hl,bc
67	43	ld b,e	bit 0,e	ld (nn),bc
68	44	ld b,h	bit 0,h	neg
69	45	ld b,l	bit 0,l	retn
70	46	ld b, (hl)	bit 0, (hl)	im 0
71	47	ld b,a	bit 0,a	ld i,a
72	48	ld c,b	bit 1,b	in c, (c)
73	49	ld c,c	bit 1,c	out (c),c
74	4A	ld c,d	bit 1,d	adc hl,bc
75	4B	ld c,e	bit 1,e	ld bc,(nn)
76	4C	ld c,h	bit 1,h	
77	4D	ld c,l	bit 1,l	reti
78	4E	ld c, (hl)	bit 1, (hl)	
79	4F	ld c,a	bit 1,a	ld r,a
80	50	ld d,b	bit 2,b	in d, (c)
81	51	ld d,c	bit 2,c	out (c),d
82	52	ld d,d	bit 2,d	sbc hl,de
83	53	ld d,e	bit 2,e	ld (nn),de
84	54	ld d,h	bit 2,h	
85	55	ld d,l	bit 2,l	
86	56	ld d, (hl)	bit 2, (hl)	im 1

87	57	ld d,a	bit 2,a	ld a,i
88	58	ld e,b	bit 3,b	in e,(c)
89	59	ld e,c	bit 3,c	out (c),e
90	5A	ld e,d	bit 3,d	adc hl,de
91	5B	ld e,e	bit 3,e	ld de,(nn)
92	5C	ld e,h	bit 3,h	
93	5D	ld e,l	bit 3,1	
94	5E	ld e, (hl)	bit 3, (hl)	im 2
95	5F	ld e,a	bit 3,a	ld a,r
96	60	ld h,b	bit 4,b	in h, (c)
97	61	ld h,c	bit 4,c	out (c),h
98	62	ld h,d	bit 4,d	sbc hl,hl
99	63	ld h,e	bit 4,e	ld (nn),hl
100	64	ld h,h	bit 4,h	
101	65	ld h,l	bit 4,l	
102	66	ld h, (hl)	bit 4, (hl)	
103	67	ld h,a	bit 4,a	rrd
104	68	ld l,b	bit 5,b	in l,(c)
105	69	ld l,c	bit 5,c	out (c),l
106	6A	ld l,d	bit 5,d	adc hl,hl
107	6B	ld l,e	bit 5,e	ld hl,(nn)
108	6C	ld l,h	bit 5,h	
109	6D	ld l,l	bit 5,l	
110	6E	ld l, (hl)	bit 5, (hl)	
111	6F	ld l,a	bit 5,a	rld
112	70	ld (hl),b	bit 6,b	in f,(c)
113	71	ld (hl),c	bit 6,c	
114	72	ld (hl),d	bit 6,d	sbc hl,sp
115	73	ld (hl),e	bit 6,e	ld (nn),sp
116	74	ld (hl),h	bit 6,h	
117	75	ld (hl),l	bit 6,l	
118	76	halt	bit 6, (hl)	
119	77	ld (hl),a	bit 6,a	
120	78	ld a,b	bit 7,b	in a,(c)
121	79	ld a,c	bit 7,c	out (c),a
122	7A	ld a,d	bit 7,d	adc hl,sp
123	7B	ld a,e	bit 7,e	ld sp, (nn)
124	7C	ld a,h	bit 7,h	
125	7D	ld a,l	bit 7,1	
126	7E	ld a, (hl)	bit 7, (hl)	
127	7F	ld a,a	bit 7,a	
128	80	add a,b	res 0,b	
129	81	add a,c	res 0,c	
130	82	add a,d	res 0,d	
131	83	add a,e	res 0,e	
132	84	add a,h	res 0,h	
133	85	add a,l	res 0,l	

134	86	add a, (hl)	res 0, (hl)	
135	87	add a,a	res 0,a	
136	88	adc a,b	res 1,b	
137	89	adc a,c	res 1,c	
138	8A	adc a,d	res 1,d	
139	8B	adc a,e	res 1,e	
140	8C	adc a,h	res 1,h	
141	8D	adc a,l	res 1,l	
142	8E	adc a, (hl)	res 1, (hl)	
143	8F	adc a,a	res 1,a	
144	90	sub b	res 2,b	
145	91	sub c	res 2,c	
146	92	sub d	res 2,d	
147	93	sub e	res 2,e	
148	94	sub h	res 2,h	
149	95	sub l	res 2,l	
150	96	sub (hl)	res 2, (hl)	
151	97	sub a	res 2,a	
152	98	sbc a,b	res 3,b	
153	99	sbc a,c	res 3,c	
154	9A	sbc a,d	res 3,d	
155	9B	sbc a,e	res 3,e	
156	9C	sbc a,h	res 3,h	
157	9D	sbc a,l	res 3,1	
158	9E	sbc a, (hl)	res 3, (hl)	
159	9F	sbc a,a	res 3,a	
160	A0	and b	res 4,b	ldi
161	A1	and c	res 4,c	cpi
162	A2	and d	res 4,d	ini
163	A3	and e	res 4,e	outi
164	A4	and h	res 4,h	
165	A5	and l	res 4,1	
166	A6	and (hl)	res 4, (hl)	
167	A7	and a	res 4,a	
168	A8	xor b	res 5,b	ldd
169	A9	xor c	res 5,c	cpd
170	AA	xor d	res 5,d	ind
171	AB	xor e	res 5,e	outd
172	AC	xor h	res 5,h	
173	AD	xor l	res 5,l	
174	AE	xor (hl)	res 5, (hl)	
175	AF	xor a	res 5,a	
176	B0	or b	res 6,b	ldir
177	B1	or c	res 6,c	cpir
178	B2	or d	res 6,d	inir
179	B3	or e	res 6,e	otir
180	B4	or h	res 6,h	

181	B5	or l	res 6,1	
182	B6	or (hl)	res 6, (hl)	
183	B7	or a	res 6,a	
184	B8	cp b	res 7,b	lddr
185	B9	cp c	res 7,c	cpdr
186	BA	cp d	res 7,d	indr
187	BB	cp e	res 7,e	otdr
188	BC	cp h	res 7,h	
189	BD	cp l	res 7,l	
190	BE	cp (hl)	res 7, (hl)	
191	BF	cp a	res 7,a	
192	C0	ret nz	set 0,b	
193	C1	pop bc	set 0,c	
194	C2	jp nz, nn	set 0,d	
195	C3	jp nn	set 0,e	
196	C4	call nz,nn	set 0,h	
197	C5	push bc	set 0,1	
198	C6	add a,n	set 0, (hl)	
199	C7	rst 0	set 0,a	
200	C8	ret z	set 1,b	
201	C9	ret	set 1,c	
202	CA	jp z,nn	set 1,d	
203	CB	—	set 1,e	
204	CC	call z,nn	set !,h	
205	CD	call nn	set 1,l	
206	CE	adc a,n	set 1, (hl)	
207	CF	rst 8	set 1,a	
208	D0	ret nc	set 2,b	
209	D1	pop de	set 2,c	
210	D2	jp nc,nn	set 2,d	
211	D3	out (n),a	set 2,e	
212	D4	call nc,nn	set 2,h	
213	D5	push de	set 2,1	
214	D6	sub,n	set 2, (hl)	
215	D7	rst 16	set 2,a	
216	D8	ret c	set 3,b	
217	D9	exx	set 3,c	
218	DA	jp c,nn	set 3,d	
219	DB	in a,(n)	set 3,e	
220	DC	call c,nn	set 3,h	
221	DD	—	set 3,l	
222	DE	sbc a,n	set 3, (hl)	
223	DF	rst 24	set 3,a	
224	E0	ret po	set 4,b	
225	E1	pop hl	set 4,c	
226	E2	jp po,nn	set 4,d	
227	E3	ex (sp),hl	set 4,e	

228	E4	call po,nn	set 4,h
229	E5	push hl	set 4,l
230	E6	and n	set 4, (hl)
231	E7	rst 32	set 4,a
232	E8	ret pe	set 5,b
233	E9	jp (hl)	set 5,c
234	EA	jp pe,nn	set 5,d
235	EB	ex de,hl	set 5,e
236	EC	call pe,nn	set 5,h
237	ED	—	set 5,l
238	EE	xor n	set 5, (hl)
239	EF	rst 40	set 5,a
240	F0	ret p	set 6,b
241	F1	pop af	set 6,c
242	F2	jp p,nn	set 6,d
243	F3	di	set 6,e
244	F4	call p,nn	set 6,h
245	F5	push af	set 6,1
246	F6	or n	set 6, (hl)
247	F7	rst 48	set 6,a
248	F8	ret m	set 7,b
249	F9	ld sp,hl	set 7,c
250	FA	jp m,nn	set 7,d
251	FB	ei	set 7,e
252	FC	call m,nn	set 7,h
253	FD	—	set 7,l
254	FE	cp n	set 7, (hl)
255	FF	rst 56	set 7,a

Tabla A3. *Códigos de registros índice. Columnas 3 y 5 se refieren al registro ix. Columnas 4 y 6 se refieren al registro iy. Todas las instrucciones que ha en esta tabla emulan instrucciones para el registro dual hl que hay en tabla A2.*

Decimal	*Hexadecimal*	*Después de 221 (Hex DD)*	*Después de 253 (Hex FD)*	*Después de 221, 203 (Hex DD, CB)*	*Después de 253, 203 (Hex FD, CB)*
6	06			rlc (ix + d)	rlc (iy + d)
9	09	add ix, bc	add iy,bc		
14	OE			rrc (ix + d)	rrc (iy + d)
22	16			rl (ix + d)	rl (iy + d)
25	19	add ix,de	add iy,de		
30	IE			rr (ix + d)	rr (iy + d)
33	21	ld ix,nn	ld iy,nn		
34	22	ld (nn),ix	ld (nn),iy		
35	23	inc ix	inc iy		

38	26			sla (ix + d)	sla (iy + d)
41	29	add ix,ix	add iy,iy		
42	2A	ld ix,(nn)	ld iy,(nn)		
43	2B	dec ix	dec iy		
46	2E			sra (ix + d)	sra (iy + d)
52	34	inc (ix + d)	inc (iy + d)		
53	35	dec (ix + d)	dec (iy + d)		
54	36	ld (ix + d),n	ld (iy + d),n		
57	39	add ix,sp	add iy,sp		
62	3E			srl (ix + d)	srl (iy + d)
70	46	ld b, (ix + d)	ld b, (iy + d)	bit 0,(ix + d)	bit 0,(iy + d)
78	4E	ld c,(ix + d)	ld c,(iy + d)	bit 1,(ix + d)	bit 1,(iy + d)
86	56	ld d,(ix + d)	ld d,(iy + d)	bit 2,(ix + d)	bit 2,(iy + d)
94	5E	ld e,(ix + d)	ld e,(iy + d)	bit 3,(ix + d)	bit 3,(iy + d)
102	66	ld h,(ix + d)	ld h,(iy + d)	bit 4,(ix + d)	bit 4,(iy + d)
110	6E	ld l,(ix + d)	ld l,(iy + d)	bit 5,(ix + d)	bit 5,(iy + d)
112	70	ld (ix + d),b	ld (iy + d),b		
113	71	ld (ix + d),c	ld (iy + d),c		
114	72	ld (ix + d),d	ld (iy + d),d		
115	73	ld (ix + d),e	ld (iy + d),e		
116	74	ld (ix + d),h	ld (iy + d),h		
117	75	ld (ix + d),l	ld (iy + d),l		
118	76			bit 6, (ix + d)	bit 6, (iy + d)
119	77	ld (ix + d),a	ld (iy + d),a		
126	7E	ld a,(ix + d)	ld a,(iy + d)	bit 7, (ix + d)	bit 7,(iy + d)
134	86	add a,(ix + d)	add a,(iy + d)	res 0,(ix + d)	res 0,(iy + d)
142	8E	adc a,(ix + d)	adc a,(iy + d)	res 1,(ix + d)	res 1,(iy + d)
150	96	sub (ix + d)	sub (iy + d)	res 2,(ix + d)	res 2,(iy + d)
158	9E	sbc a,(ix + d)	sbc a,(iy + d)	res 3,(ix + d)	res 3,(iy + d)
166	A6	and (ix + d)	and (iy + d)	res 4,(ix + d)	res 4,(iy + d)
174	AE	xor (ix + d)	xor (iy + d)	res 5,(ix + d)	res 5,(iy + d)
182	B6	or (ix + d)	or (iy + d)	res 6,(ix + d)	res 6,(iy + d)
190	BE	cp (ix + d)	cp (iy + d)	res 7, (ix + d)	res 7, (iy + d)
198	C6			set 0,(ix + d)	set 0,(iy + d)
206	CE			set 1,(ix + d)	set 1,(iy + d)
214	D6			set 2,(ix + d)	set 2,(iy + d)
222	DE			set 3,(ix + d)	set 3,(iy + d)
225	E1	pop ix	pop iy		
227	E3	ex (sp),ix	ex (sp),iy		
229	E5	push ix	push iy		
230	E6			set 4,(ix + d)	set 4,(iy + d)
233	E9	jp ix	jp iy		
238	EE			set 5,(ix + d)	set 5,(iy + d)
246	F6			set 6,(ix + d)	set 6,(iy + d)
249	F9	ld sp,ix	ld sp,iy		
254	FE			set 7,(ix + d)	set 7,(iy + d)

